DOCUMENT RESUME

ED 274 314 IR 012 291

AUTHOR	Fong, Elizabeth N.; And Others
TITLE	Guide on Logical Database Design.
INSTITUTION	National Bureau of Standards (DOC), Washington, D.C. Inst. for Computer Sciences and Technology.
REPORT NO	NBS/SP-500-122
PUB DATE	Feb 85
NOTE	118p.; Part of the series, Reports on Computer Science and Technology.
AVAILABLE FROM	Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.
PUB TYPE	Guides - Classroom Use - Guides (For Teachers) (052) -- Reports - Research/Technical (143)
EDRS PRICE	MF01/PC05 Plus Postage.
DESCRIPTORS	*Database Management Systems; *Databases; Data Collection; Dictionaries; Flow Charts; Guidelines; *Information Systems; Information Transfer; *Models; Standards; *Systems Development

ABSTRACT

This report discusses an iterative methodology for logical database design (LDD). The methodology includes four phases: local information-flow modeling, global information-flow modeling, conceptual schema design, and external schema modeling. These phases are intended to make maximum use of available information and user expertise, including the use of a previous needs analysis, and to prepare a firm foundation for physical database design and system implementation. The methodology recommends analysis from different points of view--organization, function, and event--in order to ensure that the logical database design accurately reflects the requirements of the entire population of future users. Also recommended is computer support from a data dictionary system in order to conveniently and accurately handle the volume and complexity of design documentation and analysis. This report places the methodology in the context of the complete system life cycle, and an appendix of illustrations shows examples of how the four phases of the methodology can be implemented. (Author/DJR)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. Department
of Commerce

National Bureau
of Standards

Computer Science and Technology

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it

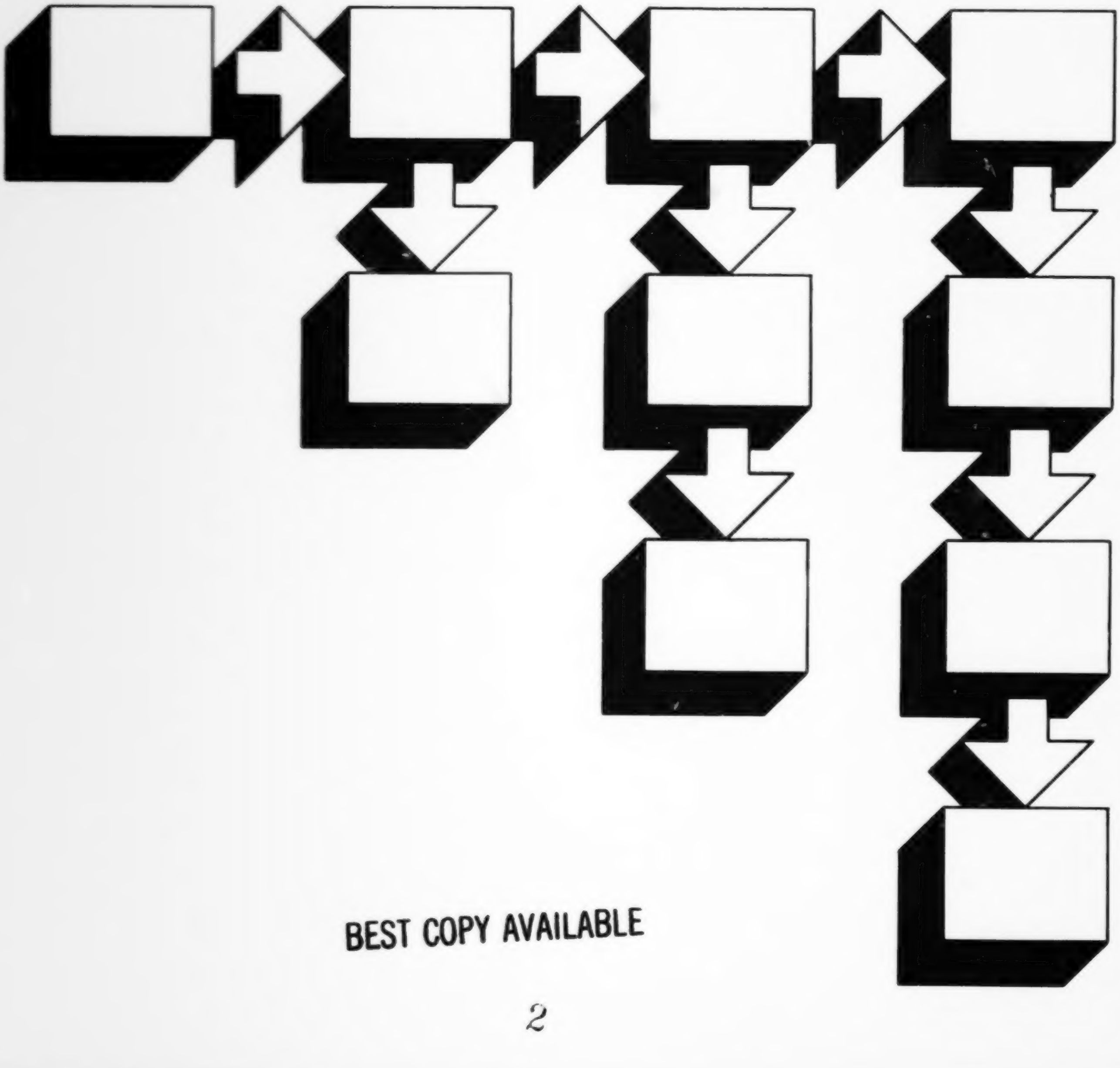
☐ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

NBS Special Publication 500-122

Guide on Logical Database Design

ED274314



BEST COPY AVAILABLE

IR012391

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Center for Materials Science.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Center for Materials Science

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Center consists of the following Divisions:

- Inorganic Materials
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

---

[1] Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.

[2] Some divisions within the center are located at Boulder, CO 80303.

[3] Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-122

Guide on Logical Database Design

Elizabeth N. Fong
Margaret W. Henderson
David K. Jefferson
Joan M. Sullivan

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued February 1985

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 85-600500

National Bureau of Standards Special Publication 500-122
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-122, 115 pages (Feb. 1985)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1985

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

Page

1. INTRODUCTION 2

    1.1 What Is Logical Database Design? 2

        1.1.1 LDD's Relation to Other Life Cycle Phases . 2
        1.1.2 Characteristics of LDD 6

    1.2 An Ideal Logical Database Design Methodology . 8

        1.2.1 LDD Practices 8
        1.2.2 Data Dictionary System 9

    1.3 Intended Audience for this Guide 10

    1.4 Purpose of this Guide 10

    1.5 Assumptions 11

    1.6 Scope of this Guide 11

    1.7 Structure of this Guide 12

2. THE FRAMEWORK THAT SUPPORTS LDD 14

    2.1 The Role of LDD in the Life Cycle 14

        2.1.1 Needs Analysis 15
        2.1.2 Requirements Analysis 16
        2.1.3 Logical Database Design 17
        2.1.4 Physical Database Design 18

    2.2 Detailed Framework for LDD 19

        2.2.1 LDD Information Requirements 19
        2.2.2 LDD Phases 20
        2.2.3 Strategies for LDD Development 23
        2.2.4 Summary of LDD Features 25

3. PROJECT ORGANIZATION 26

    3.1 Functional Roles Needed for LDD 26

    3.2 Training Required for LDD 28

    3.3 Project Planning and Management Requirements . 29

4. LOCAL INFORMATION-FLOW MODELING 30

4.1 Information Used to Develop the LIM 31

4.2 Functions of the LIM 34

4.3 Procedure for Developing the LIM 34

4.3.1 Review Need for Analysis 36
4.3.2 Determine Subsystems 37
4.3.3 Plan Development of the LIM 39
4.3.4 Develop LIM 40
4.3.5 Develop Workload With Respect to LIMs 44

5. GLOBAL INFORMATION-FLOW MODELING 47

5.1 Information Used to Develop the GIM 48

5.2 Functions of the GIM 49

5.3 Procedure for Developing the GIM 49

5.3.1 Verify the LIMs 51
5.3.2 Consolidate LIMs 52
5.3.3 Refine Boundary of Automated Information System (AIS) 54
5.3.4 Produce GIM 57

6. CONCEPTUAL SCHEMA DESIGN 58

6.1 Information Used to Develop the CS 59

6.2 Functions of the CS 59

6.3 Procedure for Developing the CS 60

6.3.1 List Entities and Identifiers 62
6.3.2 Generate Relationships among Entities 64
6.3.3 Add Connectivity to Relationships 69
6.3.4 Add Attributes to Entities 72
6.3.5 Develop Additional Data Characteristics 74
6.3.6 Normalize the Collection 75

7. EXTERNAL SCHEMA MODELING 77

7.1 Information Used to Develop the ES 77

7.2 Functions of the ES 77

7.3 Procedure for Developing the ES 78

7.3.1 Extract an ES from the CS 80
7.3.2 Develop Workload With Respect to ESs 82
7.3.3 Add Local Constraints to the ES 84

8. CONCLUSIONS 85

9. ACKNOWLEDGMENTS 86

10. REFERENCES AND SELECTED READINGS 87

7A

LIST OF FIGURES

FIGURES	DESCRIPTION	PAGE
1	Information Systems Life Cycle	5
2	Diagram of the Four LDD Phases	22
3	Local Information-Flow Modeling (LIM) Procedure	35
4	Example of a LIM	41
5	Global Information-Flow Modeling (GIM) Procedure	50
6	Example of a GIM	56
7	Conceptual Schema (CS) Design Procedure	61
8	Example of an E-R Diagram	66
9	Alternate Notation for an E-R Diagram	67
10	Replacing a Relationship with an Entity	68
11	Example of an E-R Diagram with Connectivity	71
12	Example of an E-R-A Diagram	73
13	External Schema (ES) Modeling Procedure	79

LIST OF ABBREVIATIONS

AA	Application Administrator
AIS	Automated Information System
BSP	Business Systems Planning
CS	Conceptual Schema
DA	Data Administrator
DBA	Database Administrator
DBMS	Database Management System
DD	Data Dictionary
DDA	Data Dictionary Administrator
DDS	Data Dictionary System
EKNF	Elementary Key Normal Form
E-R	Entity-Relationship
E-R-A	Entity-Relationship-Attribute
ES	External Schema
GIM	Global Information-flow Model
IRDS	Information Resource Dictionary System
LDD	Logical Database Design
LIM	Local Information-flow Model
PERT	Program Evaluation and Review Technique
QA	Quality Assurance

Guide on Logical Database Design

Elizabeth N. Fong
Margaret W. Henderson
David K. Jefferson
Joan M. Sullivan

This report discusses an iterative methodology for Logical Database Design. The methodology includes four phases: Local Information-flow Modeling, Global Information-flow Modeling, Conceptual Schema Design, and External Schema Modeling. These phases are intended to make maximum use of available information and user expertise, including the use of a previous Needs Analysis, and to prepare a firm foundation for physical database design and system implementation. The methodology recommends analysis from different points of view--organization, function, and event-- in order to ensure that the logical database design accurately reflects the requirements of the entire population of future users. The methodology also recommends computer support from a data dictionary system, in order to conveniently and accurately handle the volume and complexity of design documentation and analysis. The report places the methodology in the context of the complete system life cycle. An appendix of illustrations shows examples of how the four phases of the methodology can be implemented.

Key words: data dictionary system; data dictionary system standard; data management; data model; database design; database management system, DBMS; Entity-Relationship-Attribute Model; Information Resource Dictionary System, IRDS; logical database design.

1. INTRODUCTION

1.1 What Is Logical Database Design?

Logical Database Design (LDD) is the process of determining the fundamental data structure needed to support an organization's information resource. LDD provides a structure that determines the way that data is collected, stored, and protected from undesired access. Since data collection, storage, and protection are costly, and since restructuring data generally requires expensive revisions to programs, it is important that the LDD be of high quality. This guide describes procedures that lead to the development of a high quality LDD.

A high quality LDD will be: (1) internally consistent, to reduce the chances of contradictory results from the information system; (2) complete, to ensure that known information requirements can be satisfied and known constraints can be enforced; and (3) robust, to allow adaptation of the data structure in response to foreseeable changes in the information requirements. To fulfill these considerations, a good LDD should be independent of any particular application, so that all applications can be satisfied, and independent of any particular hardware or software environment, so that the data structure can be supported in any environment. A good LDD will ensure that modularity, efficiency, consistency, and integrity are supported in the data structure underlying the databases of the information system.

1.1.1 LDD's Relation to Other Life Cycle Phases.

LDD is closely related to the life cycle phases of Needs Analysis, Requirements Analysis, and Physical Database Design. Needs analysis and requirements analysis provide the information requirements needed to perform LDD. LDD produces data models and schemas for use in physical database design. The Physical Database Design phase receives the data structures prepared during LDD and adapts them to the specific hardware and software environment to form the internal schema of each database.

Figure 1 shows LDD's place in the life cycle and depicts the functional and data activities that can be performed in parallel. LDD can be performed in parallel to the phases of Requirements Analysis, Systems Specification, and Systems Design. The synchronized performance of these phases will assist in providing the information needed for a good LDD and will result in speeding the systems development process.

By taking a brief overview of the development of an information system, we can see how LDD is used. The life cycle of an information system should consist of the following phases:

1. Needs Analysis

   Also known as Enterprise Analysis, this phase is conducted before other work on the systems development project begins. Its purpose is to establish the context and boundaries of the systems development effort, and provide the focus, scope, priorities, and initial requirements for the target system.

2. Requirements Analysis

   The results of the Needs Analysis are carried further in this phase, which provides both the functional and the data requirements for the system under development. Requirements analysis is performed in parallel to the LDD and Systems Specification phases. Prototyping may be performed during this phase to refine requirements.

3. Systems Specification

   During this phase, the functional information provided by requirements analysis is used to produce specifications for: input and output reports that are both external and internal to the system; the functions, processes, and procedures of operational subsystems; and decision support capabilities.

4. Logical Database Design

   This phase is performed concurrently with the phases of Requirements Analysis, Systems Specification, and Systems Design. During this phase, the data requirements provided by the Needs Analysis and Requirements Analysis phases are used to perform the following iterative data modeling and design activities:

A. Local and Global Information-flow Modeling

The following are defined: data flows throughout the system; information models for each application (i.e., local) and for the entire system (i.e., global); and, data classifications, requirements, and sources for the subsystems including those for decision support. The LDD data modeling activities correspond to the functional specification activities of to the Systems Specification phase.

B. Conceptual and External Schemas

The following are defined: data structures for system-wide (i.e., conceptual) and application-oriented (i.e., external) views of the system; user views of the databases including those providing decision support capabilities; and logical database schema designs and constraints. LDD schema design activities correspond to the functional design activities of the Systems Design phase.

5. Systems Design

This phase delineates: the functional control flows using the data flows from LDD; high level and detailed system architectures; the software structure design; and the module external design (i.e., the design for interfaces among modules of code).

6. Physical Database Design

This phase produces physical data flows and the detailed internal schema for the specific hardware, software, and database implementations to be used, in order to balance maximum data storage efficiency, data retrieval performance, and data update performance. Physical database design is performed in parallel to the Implementation phase.

7. Implementation

This phase produces: logic definition for programs; module design; internal data definitions; coding; testing and debugging; acceptance testing; and conversion from the old system to the new one.

INFORMATION SYSTEMS LIFE CYCLE

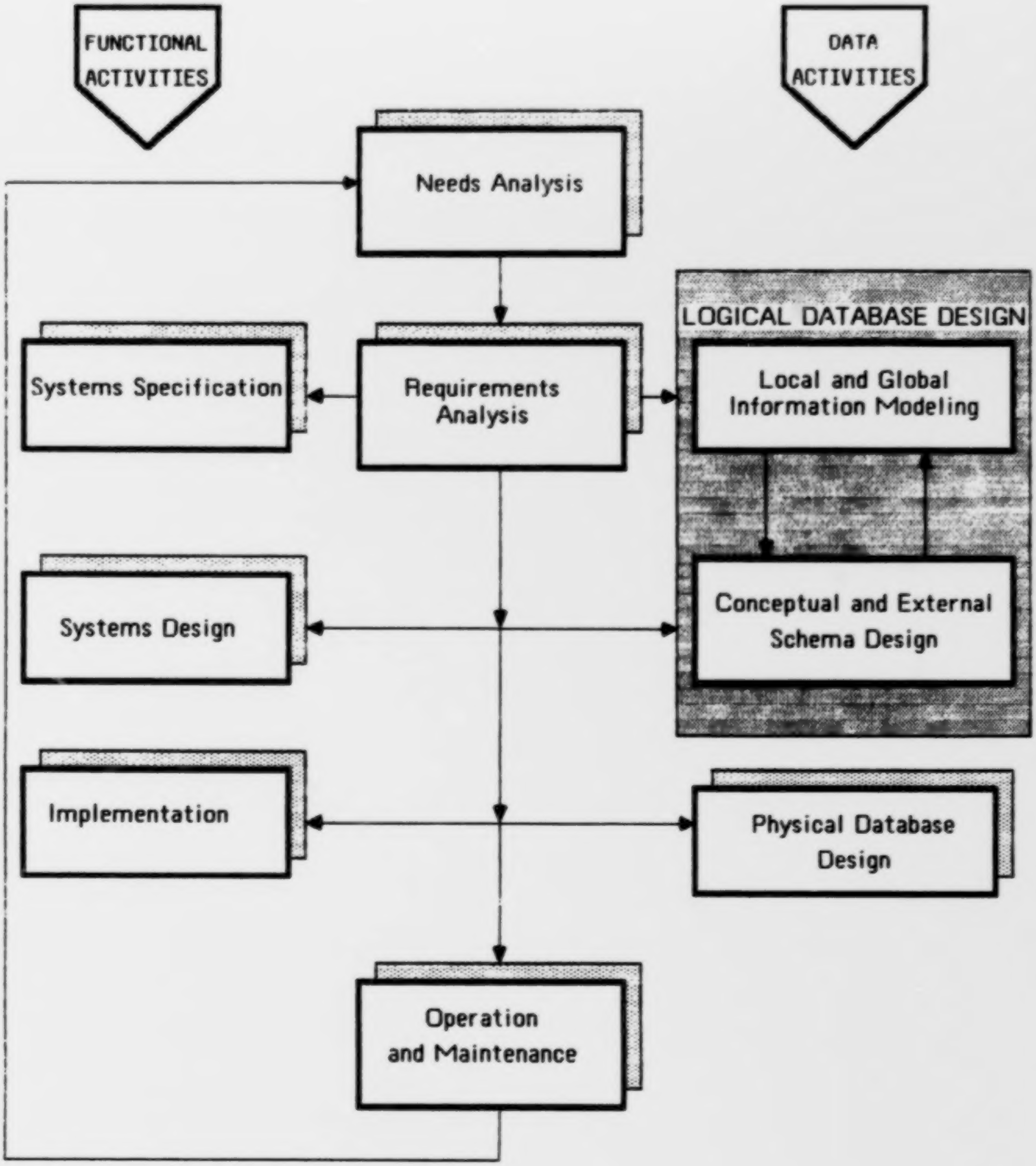

FUNCTIONAL ACTIVITIES
DATA ACTIVITIES
Needs Analysis
LOGICAL DATABASE DESIGN
Systems Specification
Requirements Analysis
Local and Global Information Modeling
Conceptual and External Schema Design
Systems Design
Implementation
Physical Database Design
Operation and Maintenance

FIGURE 1

8. Operation and Maintenance

During this phase the information system performs to serve the users' information needs and to collect data about the system's ongoing operation. Programmers and analysts continue to debug the system and modify it to support changing users' needs. Database designers continue to maintain database effectiveness and efficiency during system modifications and data changes. When modifications to the system are no longer adequate to support user needs, the current system should evolve to a new target system and the cycle will begin again.

As this description of the information system's life cycle shows, LDD plays a major role in development. LDD greatly enhances the performance of the Quality Assurance (QA) process, which would be ongoing from the Systems Specification and LDD phases through the Operation and Maintenance phase. Because LDD emphasizes the iterative approach, QA will have many opportunities to check the results of one iteration against the results of other iterations. Since LDD is performed in parallel to the Requirements Analysis, Systems Specification, and Systems Design phases, QA will be able to compare both the interim and final results of concurrent phases to resolve any difficulties sooner than through the traditional approach. The automated Data Dictionary System (DDS), described in Section 1.2.2, should be used during Requirements Analysis and LDD to provide immediate, shared access to data requirements and database designs, and to support the QA process.

1.1.2 Characteristics of LDD.

The potential benefits of LDD to the development life cycle can only be gained, however, through a good quality LDD. For LDD to perform its role well, the results of the logical design process must have certain characteristics. A LDD should be:

- o Independent of the hardware and software environment, so that the design can be implemented in a variety of environments and so the design will remain relevant even if the hardware and software selected to support the information system eventually change.
- o Independent of the implementation data model or the Database Management System (DBMS) in use, so that

the design will apply to any present or future data model or data management system, which would not necessarily be a DBMS.

- Comprehensive in representing present and future applications so that all known, anticipated, and probable needs can be included or considered in the design, to avoid costly system alterations in the future.
- Able to satisfy the information requirements of the entire organization, encompassing all possible applications rather than being limited to one or two; this way the information system will have the capacity to be an organizational resource, not just the resource of one department or application area.

A good LDD should also fulfill a set of precise technical goals to provide a firm foundation for:

- Maintainability and reusability, achieved through the use of modularity in the database design.
- Robustness, allowing both the design and the system to be adaptable to hardware and software changes.
- Security, controlled through compartmentalization in the database design which will limit specified types of data access to designated personnel or organizational units.
- Update and storage efficiency, achieved through controlled redundancy that limits the number of places where the same data will be stored.
- Retrieval efficiency, so that data can be organized to be readily accessible by system users.
- Consistency and integrity, achieved through several measures including data integrity constraints and controlled redundancy.

If done correctly, logical database design for a complex information system is a massive undertaking. The short-term cost of LDD is great, but the long-term benefits of better information and greater flexibility provide substantial savings over the system's life cycle.

1.2 An Ideal Logical Database Design Methodology

A methodology is an organized system of practices and procedures applied to a branch of knowledge to assist in the pursuit of that knowledge, which in this case is database design. In other words, a LDD methodology is a planned approach to database design that assists in database development in support of an information system.

1.2.1 LDD Practices.

This guide describes a methodology that includes the preferred practices and procedures characterizing the development of a good quality LDD and a successful information system. Although normalization is often considered the primary activity of LDD, normalization is only one of many procedures performed in LDD. Normalization is a valuable but limited tool in that it only considers functional data dependencies. Other procedures should be used in conjunction with normalization for a coherent database design. An ideal LDD methodology should be supported by:

1. A LDD guide, such as the one provided in this document, that describes clearly defined steps for analysts and designers to follow in order to produce a good LDD.

2. Analytical methods, such as the ones described in this guide, to assist in the detection of redundancies, incompleteness, and possible errors in the conceptual and functional data modeling. Some of these methods include: (a) a hierarchical, iterative approach to organizational or functional concept development; (b) differentiation of various points of view in information development, such as organizational components, higher and lower level functions, and event, control, and decision structures; and (c) normalization procedures.

3. A series of specified checkpoints for progress reviews by designers and management, and for information exchange meetings with the personnel of LDD's parallel phases, Requirements Analysis, Systems Specification, and Systems Design.

4. A mode of notation (i.e., graphic or symbolic) to describe and build a detailed conceptual model of the data and functions under study.

5. A specification language (e.g., the language used by a Data Dictionary System) to specify information requirements and the LDD design in a consistent, unambiguous manner.

6. An automated tool such as a Data Dictionary System, capable of supporting the documentation and analysis of LDD complexity, especially for large systems development projects. This tool should be used to assist in: (a) describing the conceptual model; (b) describing the data needed to support the functions of the conceptual model; (c) performing completeness and consistency checking of the conceptual model and the data needed to support the functions of the conceptual model [AFIF84].

1.2.2 Data Dictionary System.

A Data Dictionary System (DDS) is a computer software system used to record, store, protect, and analyze descriptions of an organization´s information resources, including data and programs. It provides analysts, designers, and managers with convenient, controlled access to the summary and detailed descriptions needed to plan, design, implement, operate, and modify their information systems. The DDS also provides end-users with the data descriptions that they need to formulate ad hoc queries. Equally important, it provides a common language, or framework, for establishing and enforcing standards and controls throughout an organization.

The data dictionary (DD) is the data that is organized and managed by the Data Dictionary System. The DD is a resource that will be of great value long after a logical database design is completed. The data dictionary can provide support for information about all aspects of system development to be stored, updated, and accessed throughout the system´s life cycle.

The term Information Resource Dictionary System (IRDS) is beginning to replace the term Data Dictionary System due to recognition of the flexibility and power of the software [ANSI84, FIPS80, KONI84]. This paper uses the terms Data Dictionary System (DDS) and data dictionary (DD) to conform to the current practice of software vendors.

1.3 Intended Audience for this Guide

This guide is intended primarily to provide information and guidance to: Data Administrators (DAs) and Database Administrators (DBAs) in leading their LDD projects; Applications Administrators (AAs) and application specialists in the types of data and data validation that LDD will require; and, end-users and systems analysts in how they can best contribute to the LDD project to maximize its benefits.

1.4 Purpose of this Guide

This guide provides a coherent plan of action that will allow management and database designers to direct and perform the database design successfully. The LDD plan offered here is sufficiently general to be compatible with existing tools and techniques in use for database design. By defining a methodology that provides a more stable view of the relationships among data items, this guide can be used to increase the effectiveness of an information system over its life cycle.

When the LDD approach described here is used, particularly if used with the assistance of a Data Dictionary System, an increase in clear communication can result among the end-users, systems analysts, designers, and the applications programmers who will actually code and implement the system. By providing a detailed and unambiguous description of the system's information requirements in relation to the users' perspectives, LDD offers a bridge between the end-users and the physical database designers and applications programmers.

This guide describes a methodology to be used in optimizing the flexibility and integrity of an information system. Flexibility will be ensured through the identification of the least changing characteristics of the system, which give a stable foundation upon which to build the information system. Data integrity will be optimized through the centralized control, completeness, and consistency that a quality LDD will provide. The information system that results from these LDD procedures will perform better over the system's life cycle because it will address current and probable future needs more completely and will allow requirements changes to be incorporated more effectively.

1.5 Assumptions

Several assumptions have been made in the preparation of this guide about the types of information systems in which LDD will be used. Because LDD is a non-trivial process to be undertaken when a need for it exists, it is assumed that:

- The information system's databases will be sizable and complex to support multiple applications, may have no single dominant application, and will probably contain tens or hundreds of data collections and relationships, and thousands of data elements. DBMS support is not assumed, although it is usually desirable.
- The information system and its databases are intended for use over a long period of time so that the benefits to the life cycle costs will justify the investment of time, money, and effort in LDD.
- The data requirements of the information system will be significant and include the use of ad hoc queries where the precision of the database structure will prove important.

1.6 Scope of this Guide

This guide is limited in scope to the LDD phase. The interaction of LDD with the immediately preceding and subsequent life cycle phases is mentioned, since these determine LDD's information resources and products. Because LDD works from the results of the preceding Needs Analysis and concurrent Requirements Analysis phases, and prepares a foundation for the subsequent Physical Database Design phase, these phases will be described briefly.

1.7 Structure of this Guide

Chapter 2 addresses the relationship between LDD and the phases of Needs Analysis, Requirements Analysis, and Physical Database Design. The major phases of the LDD approach are further discussed along with the types of analysis strategies that will be needed to accompany LDD. Figure 2, in Section 2.2.2, illustrates the interaction of the four phases of the LDD methodology to assist the reader in visualizing the LDD process.

In Chapter 3, the organizational aspects of the LDD project are described, including the key roles in LDD development, the training required for the personnel in these roles, and the part played by management in planning for and monitoring the LDD process.

The following chapters, 4 through 7, define the four phases of the LDD approach in detail. Chapters 4 through 7 are identically structured so that each chapter has three sections: (1) the first section of each phase discusses the information used by that phase, (2) the second section discusses the general functions of that phase, and (3) the third section discusses the procedure for accomplishing that phase. The third section of each phase includes a diagram of the steps within that phase, followed by a subsection on each step. Each step is followed by a summary chart.

Chapter 4 discusses Local Information-flow Modeling and describes three modes of analysis corresponding to the target system's (1) organizational components, (2) functions, and (3) the events to which the target information system will respond. These three analysis modes are examined in relation to data flow and data structure design techniques.

Chapter 5 addresses Global Information-flow Modeling and emphasizes the need to balance the perspectives of data flow and data structure in the development of a design that will favor both equally. The Conceptual Schema Design is described in Chapter 6 in relation to the use of Entity-Relationship-Attribute (E-R-A) data modeling diagrams and normalization techniques. Chapter 7 defines External Schema Modeling (i.e., subschema modeling) as it reflects the data structure and data flow from the end-user's perspective in the development of workload specifications for physical database design.

A glossary of acronyms used in this guide is included at the beginning of the document for reference. An appendix of examples has been included at the end of the document to illustrate the types of graphics that will be used and analysis that will occur during the four phases of LDD.

2. THE FRAMEWORK THAT SUPPORTS LDD

LDD plays an important part in the life cycle of the information system. This chapter describes: (1) the relationship between the database design and the functioning of the information system; (2) the interactions between LDD and the Needs Analysis, Requirements Analysis, and Physical Database Design phases; (3) the information requirements needed to perform LDD; (4) the phases within LDD; and (5) strategies for LDD development and their impact.

2.1 The Role of LDD in the Life Cycle

LDD defines the data structure that supports the databases of an information system. The database system and the information system are inextricably linked, but they are different.

An information system is one or more multi-purpose computer systems that may be supported by a network through which many types of users, perhaps in different locations, update, query, and provide data to the system in order to have current information available on a variety of topics. Decision support capabilities may be incorporated in the information system´s structure to assist end-users in the decision-making process.

A database is a component of an information system and may contain a variety of general and detailed information that is made available to the information system´s end-users through queries. The information system´s ability to respond to user´s queries is directly related to logical database design.

The design of the information system´s databases will determine the ways in which the information system will function. If the information system will be required to answer ad hoc queries, the data structures within the databases should be modeled to provide maximum flexibility in data accessibility and retrieval. If the system will be required to respond quickly to certain predefined queries, then the structural modeling should be constructed to support rapid retrieval performance, which will generally require indexes or redundant data. If the time and expense needed to update the data in the system are of paramount importance, then ease in locating and changing data values should be stressed in the database design. If the storage cost of large databases is a primary consideration, then the minimization of physical redundancy should be emphasized in the database design.

Usually a combination of such requirements exist for an information system, with conflicting implications for the design of the underlying databases. These requirements and their implications for the databases that support the information system are defined during the LDD phase, and their conflicts are resolved during the Physical Database Design phase.

The structure of the logical design of the database plays a crucial role in determining the capabilities and performance of an information system. A good physical database design cannot be developed without adequate preparation. A good logical database design prepares the groundwork for a quality physical database design and a successful system implementation.

The phases of Needs Analysis, Requirements Analysis, Logical Database Design, and Physical Database Design are closely linked. The ability to perform the subsequent phases is determined by the performance of the previous and parallel phases. Each of these phases must be performed well for the resulting database to represent the desired system accurately. These phases are described below.

2.1.1 Needs Analysis.

As we have seen in Chapter 1, a Needs Analysis describes the primary needs a new information system should fulfill. Without this formal expression of the organization´s perception of its needs, the analysts and designers will have to work from their own assumptions of the information system´s purposes. Their assumptions could unknowingly conflict with the organization´s vaguely described or unstated purposes. The resulting lack of clarity in direction would be costly.

A specific Needs Analysis methodology should be adopted and used by an organization previous to undertaking any extensive systems development project. The use of a well-defined methodology assures that most, if not all, of the important questions about the purpose of the proposed system will have been asked and answered at the end of the Needs Analysis phase. One of the most familiar and extensively used Needs Analysis methodologies available at this time is IBM´s Business Systems Planning (BSP) approach [MART82].

In the Needs Analysis methodology adopted, the following minimum set of questions should be posed:

1. What organizational problems require a solution that the target information system could effect?
2. What new or improved information is needed to perform what types of functions?
3. What are the boundaries and interfaces of the target system?
4. What possible improvements in information availability could be expected from the target information system? The following are goals of many system development projects:

    - Greater accuracy of information.
    - Improved timeliness.
    - Better end-user interfaces.
    - Improved privacy and security.
    - Rapid access to distant information centers by information sources and end-users.

Once a Needs Analysis methodology has been adopted and these types of questions have been answered in detail, the purposes and plans for the systems development project can be made available to the systems development personnel. If the Needs Analysis has been performed well and a comprehensive methodology has been used, sufficient information has probably been collected for LDD to begin. Close coordination with the Requirements Analysis phase is needed for LDD to continue.

2.1.2 Requirements Analysis.

The requirements analysis effort will verify and supplement the results of the Needs Analysis phase. Since LDD and Systems Specification are directly supported by the concurrent Requirements Analysis phase, it is critical that the procedures and performance of requirements analysis be planned carefully to coordinate with these other phases.

The Requirements Analysis phase will involve two types of analysis: (1) analysis of the types of data and data flows needed within the organization; and (2) analysis of the functions performed within the organization which will require the use of this data. The purpose of requirements analysis is to provide data requirements to support the LDD phase, and functional requirements to support the Systems Specification phase.

Requirements analysts verify which functions and subsystems will remain external to the system and require interfaces. By defining the information products of external subsystems or systems that are inputs to the target system, and by defining the information products of the target system that are used by external subsystems or systems, the analysts can designate the high level input/output transformations of information that must take place within the target system. The specific functions and subfunctions performed within the target system are logically organized and described. Further, the analysts define the known constraints on accuracy, timeliness, and other performance requirements, which will be further defined in LDD. Once general requirements have been described, further refinements of the requirements are developed. Prototyping may be used in conjunction with the LDD and Systems Specification phases to refine and model requirements.

As requirements are defined, the information may be stored in the form of a data dictionary to be manipulated by a Data Dictionary System. The use of a DDS will provide automated support for the storage, analysis and querying of data, for the definition and presentation of technical and management reports, and for the simultaneous access of requirements information for use in concurrent phases. Requirements information stored in a data dictionary can be supplemented with information from LDD and other phases, and can be maintained for on-line use throughout the system's life cycle.

2.1.3 Logical Database Design.

The LDD designers decide which data must be stored and maintained to support the functions and subfunctions of the target system. By abstracting from the functions to the data structures, the designer defines the data objects to be modeled and decides which properties and constraints are relevant in modeling these objects. The Conceptual Schema is the primary product of LDD.

The Entity-Relationship-Attribute modeling technique has been chosen to define the LDD data structure (see Chapter 6). Organizations that prefer other equivalent data modeling techniques may easily adapt this LDD methodology to those techniques.

An important consideration for LDD is to ensure that all information required from the LDD phase is developed and provided to the Physical Database Design phase at the appropriate time. This information required from LDD includes the volume of data, the priority and frequency of the logical access paths to be implemented in the physical database, and constraints on performance, integrity, security, and privacy.

2.1.4 Physical Database Design.

The first step of the Physical Database Design phase is to select the appropriate data model (e.g., relational, network, or hierarchical) and the data management system to support it. This selection may, unfortunately, be dictated by the software that the organization is currently using, or by the availability of software for hardware that has already been procured. Preferably, the data model and the data management system will be selected to match the requirements defined by the LDD Conceptual Schema and the workload. A useful reference in the selection process is [GALL84].

The second step, once the selection has been made, is to translate the Entity-Relationship-Attribute model from the Conceptual Schema into the selected data model. This translation is a rather simple matter for the relational model: entities become tables, relationships are implemented by means of foreign keys, and attributes become columns. The network model translation is not much more difficult: entities become records, relationships become sets or repeating groups, attributes become data items, and attributes are omitted from a member record if they are in the owner. The hierarchical model is difficult: entities become records, attributes become data items, but relationships may become either true hierarchical relationships or logical children. These translations are discussed in detail in [CHEN82] and papers referenced therein.

The next step is to develop a detailed physical data structure, including the development of indexes and other access paths, detailed record structures (perhaps combining the logical records to reduce physical accesses), loading factors, and so on. Detailed methodologies are discussed in [CARL80, CARL81, MARC78].

2.2 Detailed Framework for LDD

The information requirements needed for the performance of LDD are described in Section 2.2.1. Although LDD has previously been presented as a single phase within the information system life cycle, in Section 2.2.2 LDD will now be subdivided into four simpler phases to be performed iteratively. Strategies for analysis and the information requirements of these phases will be described in detail in Section 2.2.3.

2.2.1 LDD Information Requirements.

In addition to information obtained from Needs Analysis, LDD designers will need other information to be collected and analyzed during the Requirements Analysis phase, conducted in parallel to LDD and Systems Specification. The following information must be available to LDD designers:

- o Predefined constraints on the system, such as the use of existing hardware or software, the need to convert an existing system, and the scope of the projected information system.
- o Project constraints, such as the amount of time, money and personnel allocated by the organization for the development project.
- o Processing requirements, such as the type of functions that the information system will be expected to perform, and the general application areas that it will be expected to support.
- o Organizational, functional and data subsets, such as departments, types of actions, and types of information that the target system will be expected to supply or support.
- o Performance requirements, such as maximum retrieval and update times.
- o Capacity requirements, such as the number of data objects within the target system, and storage restrictions if the limitations of existing hardware are applicable.

- Data integrity requirements, such as the control needed over redundant data, and the need for automated integrity checks during data input and update, including edit and validation rules.

- Security and privacy requirements, such as the need for encryption for some types of data, or the limitation of access for certain types of data to specific personnel.

- Reliability and maintainability requirements that define the need for the continuous functioning of the system.

- Distributed processing and data requirements, such as the need for network connections among databases in multiple locations, or the need for shared or replicated data in multiple locations.

2.2.2 LDD Phases.

As we have seen from Chapter 1, LDD generally involves information modeling and database design that are largely hardware and software independent. LDD focuses attention on the subsystems that generate the information comprising the target system. Throughout the phases of LDD, each subsystem is examined and described in terms of: (1) the organizational components, (2) the application areas or functions, and (3) the events, which occur within or affect that subsystem. The number and type of these subsystems to be analyzed during each phase of LDD will depend on the type of analysis strategy selected, as described in Section 2.2.3.

LDD consists of four distinct phases during which all the subsystems within the system, the data flows, data structures, and user views of the databases are described. These phases are performed iteratively and in sequence until the LDD is completed. The phases of LDD are the subject of this paper and are described more fully beginning at Chapter 4. In brief, the four phases of LDD are:

1. Local Information-flow Modeling

   During this phase, data flows are modeled for individual subsystems within the target system, including each organizational component, function, and event. Subsystems are modeled one at a time. A data flow is the information that is exchanged, or "flows," within and between subsystems. Data is defined at a general rather than specific level, in terms of general formats or packages (e.g., all the data contained within a particular type of report). The products of this phase are Local Information-flow Models (LIMs).

2. Global Information-flow Modeling

   During this phase, individual data flows are combined and global data flows are modeled for collections of individual subsystems (i.e., organizational components, applications, or events) viewed as a whole. Data will continue to be viewed at the format or package level. The products of this phase are Global Information-flow Models (GIMs).

3. Conceptual Schema Design

   During this phase, the data within the data flows, defined in the previous phases, is abstracted from the packages in which it resides, and defined in terms of its functional use. The data is described in terms of: (a) entities, the basic data components; (b) relationships, the ways in which entities are associated with each other or share characteristics; and (c) attributes, the data that describes the data entities. Entity-Relationship-Attribute (E-R-A) diagrams may be used as an analysis method. The E-R-A abstraction provides the basis for a conceptual data structure. The products of this phase are Conceptual Schemas (CSs).

4. External Schema Modeling

   During this phase, the conceptual schema is adapted to conform to the needs of the application areas within the information system. By modeling the data from the user's perspective, the designer is able to verify the Conceptual Schema and derive a structured user's view of the data. The products of this phase are External Schemas (ESs) and are also known as subschemas.

Figure 2 depicts the iterative relationship of the four LDD phases. The vertical line through the center indicates a division between the phases on the left that are oriented

DIAGRAM OF THE FOUR LDD PHASES

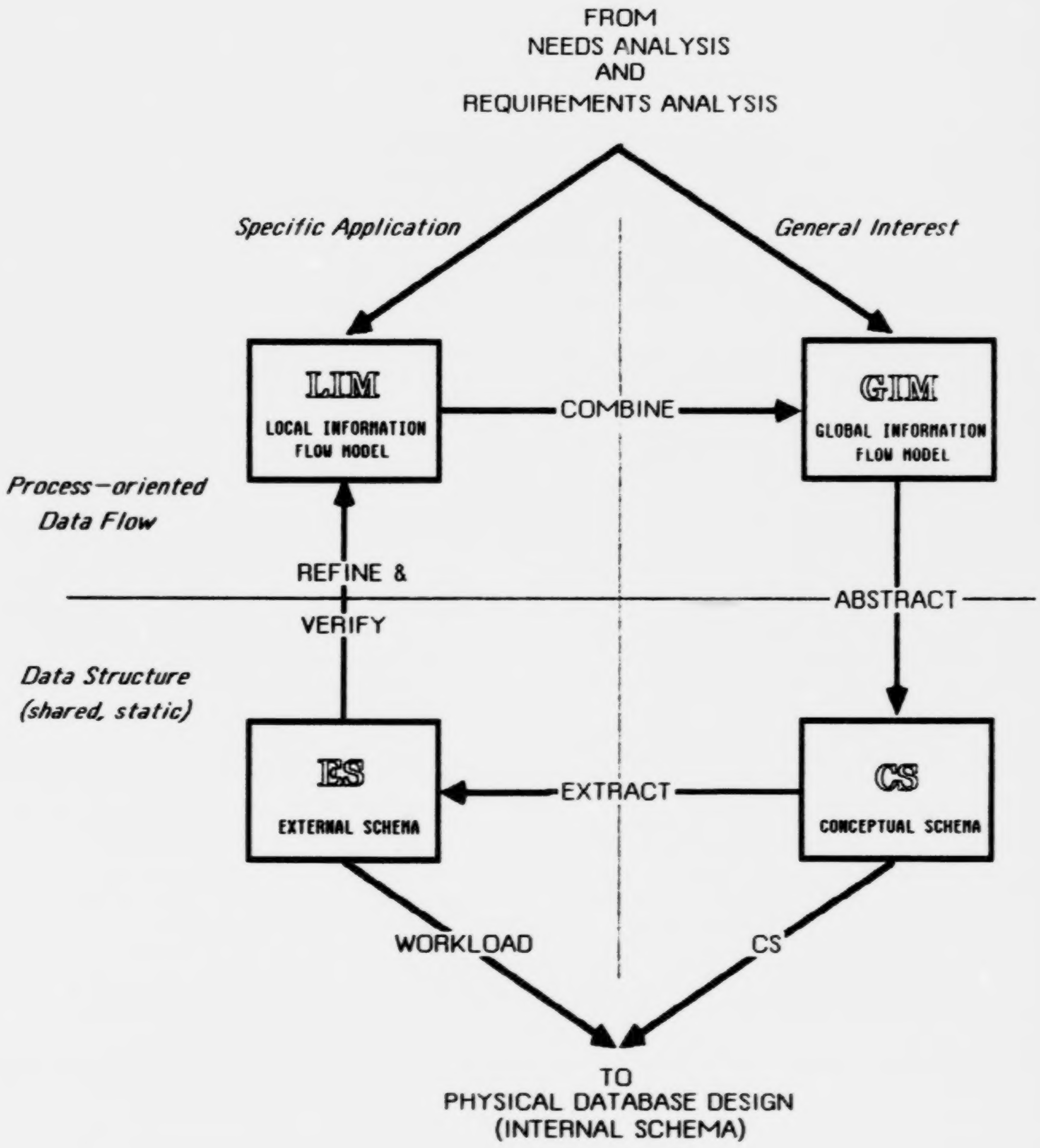

FROM
NEEDS ANALYSIS
AND
REQUIREMENTS ANALYSIS
Specific Application
General Interest
LIM
LOCAL INFORMATION FLOW MODEL
COMBINE
GIM
GLOBAL INFORMATION FLOW MODEL
Process-oriented Data Flow
REFINE &
VERIFY
ABSTRACT
Data Structure (shared, static)
ES
EXTERNAL SCHEMA
EXTRACT
CS
CONCEPTUAL SCHEMA
WORKLOAD
CS
TO
PHYSICAL DATABASE DESIGN
(INTERNAL SCHEMA)

FIGURE 2

toward a specific application (e.g., toward one organizational component, function, or event), and those phases on the right that are oriented toward organizing these specific applications into areas of general interest.

The horizontal line across the diagram indicates a division between the upper phases that are oriented toward the performance of functions and the dynamic data flow among these functions, and the lower phases that are oriented toward relatively static, shared data structures.

At the top of the diagram, Needs Analysis and Requirements Analysis indicates that these phases provide information to LDD. The results of Needs Analysis may be sufficient to begin the initial iterations of the LIM and GIM phases, particularly if the Business Systems Planning (BSP) methodology has been used. Subsequent iterations will require further information from the Requirements Analysis phase.

The diagram in Figure 2 should be read clockwise, beginning at Local Information-flow Modeling (LIM), where data flows are modeled. In Global Information-flow Modeling (GIM), the individual data flows from LIM are combined into global data flows. These are abstracted to the underlying shared entities, relationships and attributes in the Conceptual Schema (CS). Parts of the CS are then extracted to form each External Schema (ES), which is a particular user's view of the shared data. At this point, each ES is then compared with the appropriate, previously developed LIM, to ensure that the data required by the LIM has been included in the ES view. When errors are detected in this comparison, the ES, and possibly the CS, will require modification. The workload data that was originally developed for the LIM is translated into operations on data in the ES. Finally, the workload data and the CS are passed on to the next life cycle phase, Physical Database Design, for the development of the internal schema.

2.2.3 Strategies for LDD Development.

Several analysis strategies are possible in approaching LDD. The choice of the strategy will depend on the type of system to be developed and the definition of the data that will need to be integrated in its design. The scope of the data can be described as horizontal and the level of detail as vertical. The system can be viewed horizontally in the breadth of functions that the information system will support. If the system will provide many functions to many departments or locations, then the system and its data will have a broad, horizontal scope. If the system performs few functions but performs them in great detail, then the system and its data will have a depth of detail. A large system will generally include both a breadth of scope and a depth of detail. Three possible strategies for approaching the logical design phases are described, with their ramifications for system development success. Refer to Figure 2 in following the sequence of LDD procedures for the following strategies. The three strategies for approaching LDD are:

1. Breadth First.

   In this strategy, a large number of Local Information-flow Models (LIMs) will be developed at first, but in limited detail. The LIMs will then be consolidated into one Global Information-flow Model (GIM) with a broad scope but limited detail. One or more Conceptual Schemas (CSs) will be developed with broad scope but limited detail. The External Schemas (ESs) extracted from the CS will provide quality control and structure for the next iteration of LIM. The LDD phases will be repeated for the various subsystems, adding greater detail for each LIM, until the data element level is reached. This strategy is analogous to top-down system design.

   Impact: This strategy is appropriate for the development of very large, very complex information systems, where a great depth and breadth of data must be integrated through the development process.

2. Depth First.

   In this strategy, a small number of LIMs will be developed through iterations of the LDD phases to the data element level. The LIMs will be consolidated into a GIM having depth of detail but a limited horizontal scope. A small number of ESs will be developed, again with depth of detail but limited scope. Further iterations of the entire process are developed until the desired horizontal scope is attained.

   Impact: This strategy is inappropriate for the development of an information system that requires the integration of design components of considerable scope and many levels of detail. The use of this strategy may result in the need to redesign the system to effect integration. This strategy is appropriate only for the development of throw-away or expendable training or prototype projects, such as a prototype system used to verify a development concept, or an experimental system used to train personnel in other systems development concepts or in Data Dictionary System use.

3. Critical Factors First.

   In this strategy, a large number of LIMs are developed, including details for the critical aspects of the target system (e.g., critical functional requirements, critical performance characteristics, proof of concept, etc.). The LIMs will be consolidated into a GIM with broad scope but uneven detail. One or more CSs will be developed with the same broad scope but uneven levels of detail. The process will be repeated with increasing levels of detail for each LIM, with subsystems analyzed in order of priority, until the data element level is reached. The critical subsystems will be processed through the LDD cycle first, and the non-critical subsystems will follow later.

   Impact: This strategy is appropriate for the development of a very large system if the critical factors of the target system can be identified and accepted. It is also appropriate for prototype development and for evolutionary development, where some functions will be implemented first and other functions will follow.

2.2.4 Summary of LDD Features.

The four phases of LDD use a variety of symbologies to assist in analysis. These include the use of bubble diagrams in the analysis of data flows, Entity-Relationship-Attribute (E-R-A) diagrams in CS development, normalization analyses where applicable, and Data Dictionary System (DDS) contents and automated analysis reports throughout LDD.

The outputs of LDD's phases are: Local Information-flow Models (LIMs) and Global Information-flow Models (GIMs) that model data flows for the organizational components, functions, and events; Conceptual Schemas (CSs) that provide an E-R-A model, or another type of data model, for use by programmers and designers; and External Schemas (ESs) that present an application-oriented user view for use within the organization as a representation of the data to be included in the target system.

3. PROJECT ORGANIZATION

For LDD to be performed successfully, plans should be made to support the information requirements of LDD and to incorporate LDD roles into the organization. In this chapter, LDD functional roles, training, and project planning needs are described.

3.1 Functional Roles Needed for LDD

The following functional roles are described in terms of the development of LDD. A role may be performed by many people, or one person may perform several roles, depending on the complexity of the database. Some LDD roles may overlap with roles to be performed in Requirements Analysis and other phases. The roles required for LDD are the following:

- Application Administrators (AAs) who will work with designers and analysts to define and validate the data and functions. One or more AAs may be needed according to the size of the system and the complexity of the application areas. AAs will work with a number of application specialists.
- Application Specialists who are knowledgeable about the application data being modeled, or about the application functions that use the data, or about both. The application specialists will assist the designers and analysts in preparing an accurate LDD.
- Data Administrator (DA) who will facilitate the LDD and systems development process by ensuring consistency in data definition, and overseeing the data management, data integrity, and data security functions performed in LDD development. The DA will continue to perform this role in regulating these facets of the information system once it is completed, and so will also use the LDD once it is developed. The DA may have a sizable staff, depending on the complexity of the data resource and the time available to perform LDD and other tasks. The DA staff may include the Database Administrator and the Data Dictionary Administrator. The DA staff will work closely with the AAs.

- Database Administrator (DBA) who will control the database and the DBMS, facilitate the LDD and systems development process, assist in data maintenance, and use the LDD as it is developed. The DBA is concerned primarily with technical aspects of the database, in contrast to the DA, who is more concerned with information policy and interacts with management and users. The DBA will continue in this role once the information system is operational. The DBA may have a small staff to support this function. This function will continue throughout the life cycle of the target system.

- Data Dictionary Administrator (DDA) who will oversee the operation of the Data Dictionary System (DDS), and assist in the data maintenance process for LDD. The DDA may be supported by a staff, including a Librarian and possibly data entry personnel. Data entry may also be performed directly by designers and analysts in the course of their work. The DDA function should continue throughout the life cycle of the target system, to continue to maintain documentation about the system.

- Data Dictionary Librarian who will maintain the data in the data dictionary (DD), and support the LDD and systems development effort.

- Database Designers/Analysts who will develop the information requirements, logical database diagrams, models and schemas. They will be expert in database design, familiar with the DDS, and become familiar with the application areas. They will perform the functions that are the focus of this report. Database designers will be needed throughout the life cycle of the information system, to maintain high performance and efficiency as the database changes through time.

- Project Managers who will direct the LDD and systems development projects. They will be familiar with the application areas, computer systems, systems development practices, and become familiar with LDD procedures.

- End-users of the DDS and the information system under development who will access and update information in the databases, and who will generate reports and decisions from this information. End-users will include personnel from all organizational levels and will perform the following roles:

- Data Entry and Update
- Data Retrieval
- Data Analysis
- Data Management and Control
- Project Management
- Upper Management

3.2 Training Required for LDD

The personnel involved in the LDD phase of development, particularly AAs and Application Specialists, will require training so that they will be able to work with database designers as a team. Some personnel will already be knowledgeable in these areas, but many will need to be trained. Project management should arrange to have LDD personnel trained in:

- o The purpose and general procedures of LDD.
- o The points of view to be represented within the system (i.e., organizational components, functions, and events).
- o Use of the symbology, such as how to construct and interpret E-R-A and bubble diagrams.
- o Use of the Data Dictionary System or other automated tool.

End-users who review the LDD may require any of three levels of training in the use of the Data Dictionary System, depending on the extent of each end-user's responsibility:

- o Reading knowledge of LDD reports that are generated via the DDS, to be able to recognize when the report indicates a modeling error.

- Interpretive capability to understand LDD reports generated via the DDS, to be able to recognize what is wrong in a report that indicates a modeling error.
- Expert knowledge of the DDS procedures and an understanding of the products of LDD, to be able to correct errors in modeling detected in DDS reports.

3.3 Project Planning and Management Requirements

The systems development Project Manager and the LDD Manager should plan for and control the systems development project so that a high quality LDD results. In addition to the activities of traditional management roles, managers in these positions must determine that several procedures have been adopted before the project begins.

The Project Manager must be sure that good methodologies have been selected or developed for the Needs Analysis, Requirements Analysis, LDD, and other phases. In addition, it is necessary to determine that these methodologies are coordinated according to a schedule so that the results of previous and parallel phases are available for use by other phases. The schedule should also include various types of training for personnel working on parallel phases. Further, the Project Manager must decide on a strategy for LDD development that will support the breadth of scope and depth of detail to be encountered in analyzing the target system.

The Logical Database Design Manager will fill a similar role for the LDD phase. The LDD Manager will: (1) select a good LDD methodology and analysis strategy suitable to the type of system under development; (2) coordinate LDD training with the managers for parallel phases; (3) coordinate LDD activities with the Requirements Analysis Manager, so that information will be available for LDD to conform to appropriate schedules; (4) define checkpoints to review the progress of the LDD work; (5) determine the types and characteristics of the DDS documentation and analysis reports to be generated to support the LDD phases; and (6) manage the synthesis and integration of information from many sources within the organization to support LDD.

4. LOCAL INFORMATION-FLOW MODELING

A Local Information-flow Model (LIM) is a description of the movement of data collections such as reports, forms, memos, messages, transactions, and files to, from, and within a particular focal point. The focal point may be an organizational component (e.g., the personnel department), a function or application (e.g., payroll processing), or an event (e.g., a milestone in the budget cycle). The first iteration of this phase will produce a single LIM summarizing the inputs and outputs of the entire organization served by the database being designed. During subsequent iterations multiple LIMs will be produced, each describing a part of the next higher-level LIM. The level of detail may be very high (e.g., very general types of data going into or out of an entire organization), intermediate (e.g., reports and other data going into, out of, or processed within an office), or very low (e.g., transformation of an employee number into an employee name), depending on the number of iterations through the four phases of logical database design.

There are two reasons for choosing this approach:

1. Complexity is controlled at every stage of the iteration by restricting the scope of each LIM. Interviews with users can concentrate on the most critical aspects of the user's organization, function, or event, with the assurance that a higher-level context has already been developed and that details can be filled in later. The interviewer need not be overwhelmed with trying to understand everything all at once. Note that a top-down approach is advisable--starting from data elements and working up is more likely to end in a disastrous lack of direction and an abundance of confusion.

2. The different aspects--organization, function, and event--represent the fact that organizational structures are important, but they do not give a complete model of information processing. Functions and responsibilities are shared by sequential or simultaneous access to and transformation of data. All aspects may be required to give a true picture of database requirements. Note that manual functions should be analyzed if there is a significant chance that they will be automated during the life of the database.

The general objective is for a LIM to represent whatever an application specialist knows about his or her job and organization. The LIM does not represent details about how information is captured or derived before it reaches the application specialist or how it is used or processed after it leaves her or him.

The emphasis of the LIM should be on business functions and events--that is, data, operations, and products that are basic to achieving organizational objectives--rather than on any particular technology for implementing those functions. One reason for this particular emphasis is the fact that technology changes much more rapidly than the business functions (the need for payroll is constant, but the policies and technologies implementing it are changeable). A database should be relatively stable and retain its value over a long period of time--the time and cost of data collection and organization are too great to permit the database to be considered anything less than a major capital investment. Another reason for the emphasis on business functions is that these are familiar and well-understood by the data users, who are the people responsible for achieving organizational objectives. The abstract concepts of data modeling, introduced in the phase concerned with the development of the Conceptual Schema, are generally not meaningful to the user unless there is some familiar context of business functions. One way of viewing the LIM is that it is a means for relating the abstract External Schema (a part of the Conceptual Schema) to a concrete business context.

4.1 Information Used to Develop the LIM

Information that is relevant to the development of the LIM may be obtained through examination of documents or through interviews, or, preferably, through interviews based on thorough preparation via documents. The following information is generally needed:

1. The nature, objectives, structure, and scope of the subsystem must all be analyzed to ensure compatible LIMs. Both the present and the future should be considered. Non-routine operations, or operations that are performed infrequently, may be particularly important--for example, end-of-year accounting operations may have unique but critical requirements. Interactions with customers, vendors, and other parts

of the external environment may be very important.

2. Existing automated systems and other available hardware, software, and data resources should be studied to determine how they interact with the subsystem being studied; the emphasis should be on the queries, reports, and transactions that are actually relevant rather than on what is currently produced. It is important to maintain continuity with the present while still ensuring sufficient flexibility for long term growth of the information resource. Existing systems may already have replaced certain functions and as such should themselves be "interviewed." This can be difficult since existing systems may be poorly structured and documented. However, existing systems have already solved problems -- what are those problems? Existing systems may be enforcing policies that the people are no longer aware of -- what are those policies? Existing systems may also be creating data that everyone takes for granted -- how are existing systems combining files, applying algorithms, etc.?

3. The subsystem's perspective on decisions must be analyzed. The position titles and descriptions held by decision-makers, the business models that they use, the information that they require, and the relationships that they have with other decision-makers must all be analyzed. Senior management views (strategic planning), middle management views (control and tactical policy), and applications views (operations) are all required to give balance to the total collection of LIMs. Historical and "what if" data are particularly important in analyzing the data flow of higher-level decision makers.

4. Real-world rules and policies should be studied. Geographic location requirements are particularly important (e.g., there is little point in designing a highly integrated central database if the policy is to maintain local control of data). Policies on data retention and archiving may also be important (e.g., archiving may constitute a major information subsystem). Security, privacy, integrity, and error handling policies (including policies and procedures for recovery from both data processing and organizational mistakes) may have major effects on the data structures (for example, classified and unclassified data may have to be stored separately).

5. A catalog of reports and forms needed for routine tasks is clearly relevant to the LIM. Collections of reports and forms are relevant to high-level LIMs, individual reports and forms are relevant to intermediate-level LIMs, and parts of reports and forms are relevant to low-level LIMs. The timeliness and quality of the reports and forms should be recorded. Reports that have outlived their usefulness are irrelevant to LDD.

6. Collections of informal data are also very important. This data can include files or folders of memos and letters (e.g., Freedom of Information Act requests, and customer complaints in writing), notes on telephone conversations (e.g., payroll inquiries), and databases on personal computers.

7. Formal reference data collections such as FIPS codes, ZIP codes, pay scale tables, and address or telephone directories are relevant.

8. "Log" books or lists may be used to assign unique numbers, organize office functions, record significant events, or otherwise coordinate activities.

9. Other regular sources of information, such as telephone contacts, should be carefully studied, since these may be very relevant to getting the job done.

10. Information from the higher-level GIM and the higher-level LIM which is being subdivided provide context for developing more detailed LIMs in successive iterations of the LDD cycle. Once LDD has begun, the examination of this information will be the first step in providing a LIM.

11. Quantitative information on volume of data and frequency of processing for all of the above. This information will be used to help develop an estimate of the database workload.

Since each LIM is a refinement of the previous iteration of the design cycle, the LIM is constrained by the previous higher-level LIM and External Schema. If deeper analysis uncovers an error at the higher level, then that higher-level should be corrected before proceeding further. Otherwise, other lower-level LIMs, based on the erroneous LIM and External Schema, may contain errors or be inconsistent with each other.

4.2 Functions of the LIM

The primary function of the LIM is to serve as part of the Global Information-flow Model (GIM). Other functions of the LIM are:

1. The LIM provides a guide for the development of further details. Each iteration is based on a decomposition of a previously developed LIM, unless the focus is switched from an organizational component to a function or event, in which case the new LIMs are based on combinations of previously developed LIMs.
2. The LIM may be used as a guide to planning the development of a new application program or system, modifying an old application program or system, or modifying the organizational structure. In each case, the LIM is analyzed to see whether the flow of data is efficient and effective; changes are suggested if unused reports are being produced, if similar functions are being performed unnecessarily, if functions that should be performed by a computer system are being performed manually, or if the data flow can be reduced by combining organizational components that sequentially process the same data.
3. The LIM is also used to collect information concerning the database workload. This information is eventually used to optimize and evaluate the physical database design.

4.3 Procedure for Developing the LIM

Figure 3 shows the five sequential steps in the development of the LIM. The steps are described in the following paragraphs.

LOCAL INFORMATION-FLOW MODELING (LIM) PROCEDURE

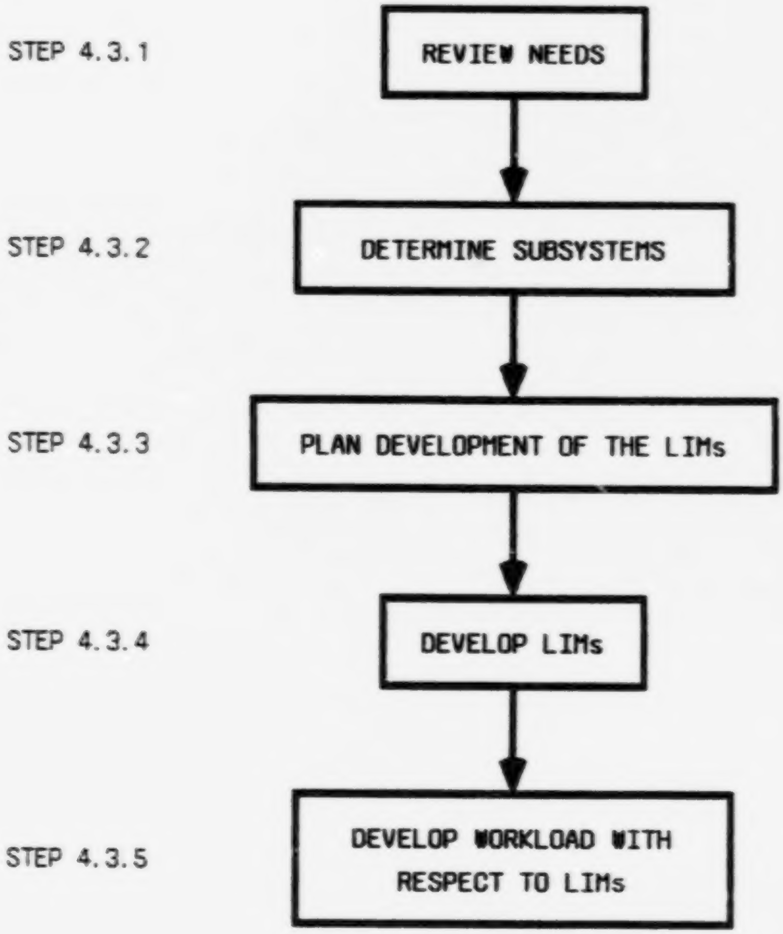

STEP 4.3.1
REVIEW NEEDS
STEP 4.3.2
DETERMINE SUBSYSTEMS
STEP 4.3.3
PLAN DEVELOPMENT OF THE LIMs
STEP 4.3.4
DEVELOP LIMs
STEP 4.3.5
DEVELOP WORKLOAD WITH RESPECT TO LIMs

FIGURE 3

4.3.1 Review Need for Analysis.

The primary function of this step is to determine whether the organizational component, function, or event under consideration should be subdivided for further analysis, or whether it has already been analyzed sufficiently.

The first iteration of the logical database design methodology will begin with a preliminary determination of boundaries--that is, which organizational components, functions, and events require interaction with the proposed database. Next, it is necessary to determine the best method for subdividing the design problem--by organizational components, by functions, or by events. Generally, the first few subdivisions will be along organizational boundaries. These boundaries are usually well-defined, familiar, and non-threatening to the application specialists. They serve very well in identifying broad classes of data, major functions and events, and data flows.

Organizational decomposition may be insufficient, however, for the detailed development of data structures which are shared among different organizational components. Later iterations should concentrate on subdividing the functions and events that have been identified during the study of organizational subdivisions; such functions and events must provide data to the database and use data from it, so are directly relevant to the structure of the database.

Since functions and events frequently cross organizational boundaries, their analysis may suggest the need for reorganization to eliminate duplicate or unnecessary jobs, and will almost certainly require cooperation among application specialists from different organizational components. Consequently, such analysis is very delicate and should not be attempted too early in the LDD process.

Eventually it will be determined that there is no need to subdivide any more functions or events; the logical database design process is then "complete," although maintenance of the LIMs and other products must continue indefinitely.

Step 4.3.1 Review Need for Analysis	
Function:	To determine whether more detail is required
Output:	Determination of whether to subdivide a subsystem
Team Members:	User - AA, DA Developer - AA, DA
Tools:	Use DD to report on previous work
Guidelines:	Decision involves both technical and management issues

4.3.2 Determine Subsystems.

Once a decision has been made to subdivide an organizational component, function, or event, the next step is to determine the appropriate subdivisions. Two situations may be distinguished:

1. The subdivision involves a further refinement of an organizational component, function, or event. This is the normal case in business systems analysis, so various methodologies from business systems planning, organizational analysis, and software engineering may be applied. Either function-oriented methodologies [DEMA78, GANE79, MYER78, ROSS77] or data-oriented methodologies [JACK83, ORRK82] may be used as measures of the relative merit of different decompositions.

2. The subdivision involves a switch from one type of analysis to another. For example, the previous iteration of subdivision was based on organizational components, but this iteration is to be based on functions. In this case, the primary activity is composition, rather than decomposition--the various aspects of a function that appear in different organizational components must first be joined together to form a coherent statement of the whole function, and

then functional decomposition can proceed at later iterations. Clearly, it is extremely important that data flow has been carefully documented during previous iterations; data flow is the primary clue to the common basis for different organizational perspectives on a single function. The effect of a Data Dictionary System is to allow the DA to combine an organizational hierarchy, a functional hierarchy, and an event hierarchy into a consistent network which can be supported by the database structure.

In either case, the result will be a list of well-defined subsystems--organizational components, functions, or events-- of the LIM being analyzed. The subsequent steps will determine how each subsystem interacts with the data flowing into or out of that LIM, and the data flowing from or to the other subsystems.

Step 4.3.2 Determine Subsystems

Function:	Determination of how to subdivide a subsystem
Output:	List of lower-level subsystems
Team Members:	User - AA, DA Developer - AA, DA
Symbology:	Organization charts, data-flow or event diagrams
Tools:	Use DD to represent organizational components, functions, or events
Guidelines:	Care is required -- poorly chosen subsystems will have overly complex interfaces

4.3.3 Plan Development of the LIM.

This step involves the development of a detailed plan for this iteration of the analysis. The plan may include priorities, so that decomposition will consider critical factors first. Two strategies are possible:

1. Each step in the subdivision spawns a set of independent plans. Detailed work may proceed in parallel, given a sufficiently large staff, with the results coordinated primarily through the data dictionary. The advantage of this approach is that planning is minimized. The disadvantage is that quality control of the data dictionary becomes extremely critical during and after execution of the plan. Synonyms and homonyms for functions and data must be detected and resolved quickly or different analysis paths will unknowingly overlap, resulting in confusion and duplication of effort. The philosophy of this strategy is to move quickly and solve problems later (possibly during the development of the GIM).

2. Each step in the subdivision involves the development of a single, coordinated plan. Detailed work is coordinated in advance, so that problems of synonyms, homonyms, and duplicated effort are minimized. The advantage of this approach is that overall control of the effort is maintained. The obvious disadvantage is that this approach requires extremely knowledgeable DA and AA staff to formulate, monitor, and control the execution of the plan. Also, more work must be done serially rather than in parallel.

In either case, it is necessary to develop a detailed project management plan, with milestones, time and cost estimates, and assignments for application specialists as well as for AA and DA personnel.

Step 4.3.3 Plan Development of the LIM

Function:	Develop project management plan for this subsystem
Output:	Milestones, time and cost estimates
Team Members:	User - AA, DA Developer - AA, DA, Managers
Symbology:	Project management charts
Tools:	Use DD to represent project management data and boundaries
Guidelines:	Assignments must be very specific

4.3.4 Develop LIM.

Various system analysis and design methodologies may be used in conjunction with a data dictionary to document the data flows that are developed. Either function-oriented methodologies [DEMA78, GANE79, MYER78, ROSS77] or data-oriented methodologies [JACK83, ORRK82] are suitable. Whereas previous steps have involved consultation with management, this step and the following are best accomplished by short interviews (no more than two hours per iteration) with application specialists. Reference material and the LIM developed during the previous iteration are used to prepare for the interview and to verify the analyst´s interpretation of the application specialist´s statements. All materials may be made available to the application specialists in advance of the interview. (Note that discrepancies revealed during an interview should prompt further questions rather than challenges--the interview should not be threatening.) Graphical simplicity is very desirable, so that untrained users can judge the correctness of the LIMs that are relevant to them.

Useful types of diagrams include the following:

1. An organization chart can be used to show the hierarchical relationships among organizational LIMs.

2. A "bubble" diagram with an organizational focal point connected to other organizations by data flows can be used to represent an organizational LIM, as in the following:

EXAMPLE OF A LOCAL INFORMATION-FLOW MODEL

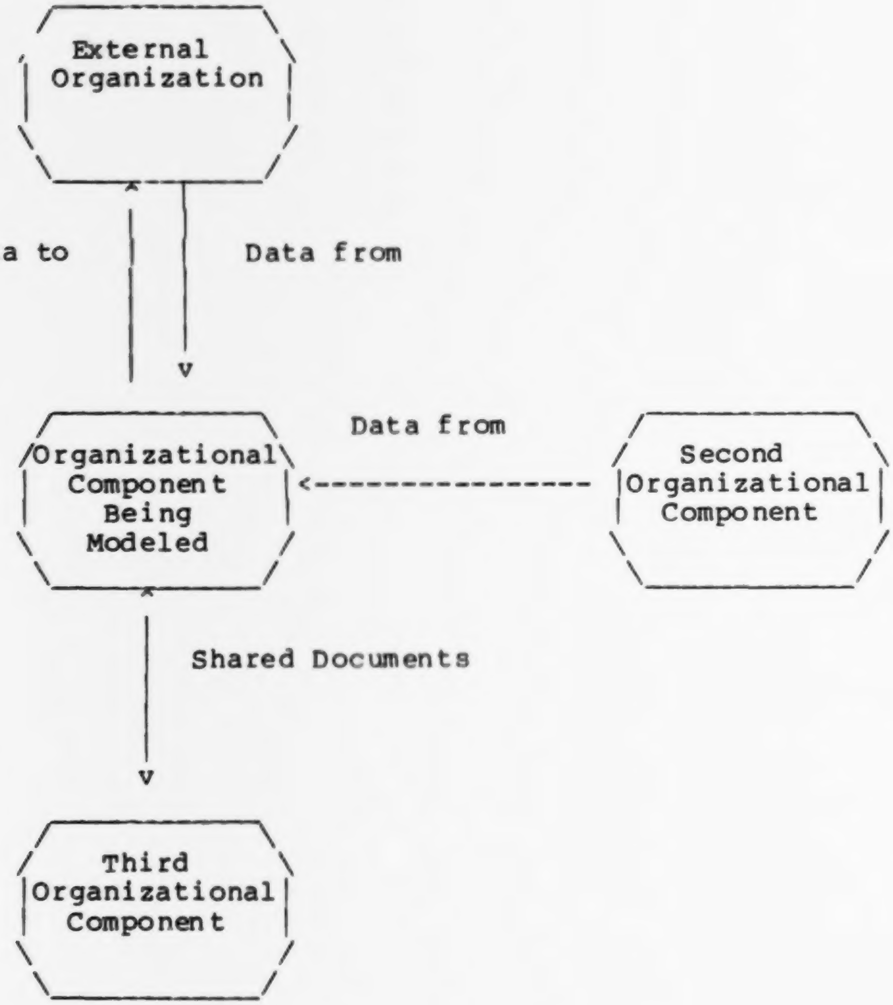

External Organization
Data to
Data from
Organizational Component Being Modeled
Data from
Second Organizational Component
Shared Documents
Third Organizational Component

Figure 4

3. A functional hierarchy can be used to show the hierarchical relationships among the functional LIMs.

4. A data-flow diagram [DEMA78, GANE79, MYER78] or action diagram [ROSS77] can be used to show inputs, outputs, subfunctions, and data flows among the subfunctions of a functional LIM. (Note that this type of diagram shows two levels of the LIM hierarchy.)

5. A Gantt chart can be used to show the temporal relationships among events.

6. A PERT chart can be used to show the relationships, especially time dependencies, among functions and events.

7. A state-vector diagram [JACK83] or a decision table can be used to show additional details of functions and events.

The data dictionary is used to record detailed information that would only confuse a diagram; automated analysis of program code, job control language, and audit trails may provide much of the detail. The selectivity of data dictionary queries and reports helps to make the details comprehensible. Diagrams should be produced automatically from the data dictionary. Also, graphic input could be a means of populating the data dictionary when this capability becomes automated in the future.

A special but important example of data flow is storage and retrieval of information by an organizational component, function, or event; the storage medium is treated like another organizational component, function, or event.

Data flow is used to determine the formal consistency and completeness of the analysis--for example, whether each data flow has a source and a sink (either may be some internal storage medium). The use of a data dictionary is extremely important in this situation to ensure that all of the various aspects of the function are considered. The views of all users who interact with a function must be reflected in that function.

The description of data flows should generally include one level of decomposition. For example, if the data flows in a top-level functional analysis are collections of reports, then each data description in the data dictionary should include a list of the component reports. At a lower level, if the data flows are reports, then their descriptions should include subdivisions of the reports--selected columns, or rows between subtotals, or the subtotals themselves, for example. At a very detailed level, the data descriptions would be data elements.

Information which is useful in understanding the relative importance of the functions and in planning the next iteration of this phase includes the following:

1. Staff time, in work-years or other convenient unit, expended on performing the function.
2. The number of staff personnel performing the function.
3. The number of locations where the function is performed.
4. Whether there is a single step that consumes 80% or more of the time spent on the function.

Step 4.3.4 Develop LIMs

Function: Provide guidance to the development of the GIM and CS

Output: LIMs

Team Members: User - AA, DA
Developer - AA, DBA

Symbology: Use bubbles to represent organizational components, events, functions, or external interfaces. Use lines to represent data flows.

Tools: Use DD to represent subsystems and interfaces

Guidelines: Graphical simplicity is desirable
Use selectivity of DD reports
Should be easy for users to understand and critique

4.3.5 Develop Workload With Respect to LIMs.

The primary function of this step is to develop a preliminary description of the workload: the frequency, sequence, and selectivity with which functions use or produce data, and the volume of stored data [JEFF82, SUST84]. The workload will be used during the development of the External Schemas to determine whether the Conceptual Schema can support the LIM, and what paths must be taken through the Conceptual Schema to obtain the data required by the LIM. It will also be used to determine whether certain functions should be automated. The workload must be used during the development of the Internal Schema (physical database design) to determine appropriate physical record structures, record placement in areas, access methods, loading factors, indexes, and other parameters. Accordingly, this step must be performed during the most detailed iteration of functional analysis; it may be performed at earlier steps to provide additional quality control for the LIMs and Conceptual Schema.

At this phase, the workload is described in terms of data collections that may be very different from the logical records that will eventually constitute the final Conceptual Schema. In particular, the level at this phase may be very high (e.g., data objects like "employee," "project," and "part" rather than data elements like "employee-first-name," "estimated-project-cost," and "part-quantity-in-warehouse") and the grouping of data may be quite arbitrary (e.g., "employee" may include data about skills, projects, and organizations associated with the employee). Eventually these data objects will be restructured to form a database, so it is important to be able to map this preliminary workload into appropriate paths through that database.

The information to be collected and stored in the data dictionary should include the following:

1. The volume (number of instances) of each data collection (e.g., the number of employees, projects, and parts).
2. The priority of the function (e.g., "an airline reservation must be confirmed within 20 seconds" and "a marketing analysis on advance reservations must be available within 2 hours of a request").

3. The frequency of execution of the function.

4. The sequence with which data collections are accessed by the function, and the source of the data from input or database (e.g., start with "employee," then access "project," then access "project-manager" to determine who "manages" a given employee).

5. The parts of each data collection that are used to decide whether a given instance of that data collection is relevant (e.g., "employee-name" identifies the required "employee" data).

6. For each of the parts of data collection, the number of relevant instances (e.g., "1").

7. For each relevant data collection accessed by the function, the parts that are needed for retrieval by the function (e.g., "employee-project" is the only retrieved part of the "employee" data). If applicable, the preferred order is desirable (e.g., the "employee-project" data is to be sorted by "project-number").

8. The parts of each relevant data collection that are needed for update by the function (e.g., "employee-hours" is the only updated part of the "employee" data).

9. At each point where the function branches, the fraction of the time each branch is taken (e.g., 90% of the time "employee-project" will be non-null, so "project" will be accessed, and 10% of the time it will be null so the path will terminate).

Step 4.5.3 Develop Workload with Respect to LIMs

Function: Develop preliminary specifications for physical design

Output: LIMs with volume, frequency, sequence, and selectivity

Team Members: User - AA, DBA
Developer - AA, DBA and Analysts

Symbology: LIM diagrams

Tools: Use DD to store workload information to be used for physical design

Guidelines: Keep the scope limited to a single application

5. GLOBAL INFORMATION-FLOW MODELING

A Global Information-flow Model (GIM) is basically an interconnected collection of all of the Local Information-flow Models (LIMs). Its structure is quite complex: it combines up to three hierarchies of LIMs (a hierarchy based on organizational components, another based on functions, and possibly another based on events); these must be interconnected in terms of data flow, which itself may be a complex network of data objects, as well as other interrelationships such as organizational authority and responsibility. A Data Dictionary System (DDS) is strongly recommended to manage the GIM. In an extremely complex situation, where even a DDS is unable to present the mass of information in a meaningful way, multiple GIMs may be developed, each representing a major subsystem loosely connected to the other GIMs. Note, in particular, that the GIM, like the LIM, must generally represent both automated and manual data, and both current and planned functions.

The major task involved in developing the GIM is simply adding the new details represented by each new LIM. The new LIMs must be verified for consistency with higher-level LIMs, names must be reconciled with existing names, and the different perspectives (organization, function, and event) must be interrelated. These are basically responsibilities of the DA with assistance from the AAs in detecting and resolving potential problems in performance, cost, reliability, security, and the like. The DA should not require direct access to the users.

The GIM may be represented in various forms according to the methodology chosen. A diagram may consist of ovals or rectangles representing the subsystems, and labelled lines representing the data flows. This is a simple source-sink model which is very useful for communicating with users. Other representations of the GIM include many different types of matrices showing the interactions of organizational components, functions, events, and data objects with each other [MART82]. A data dictionary is recommended for the primary means of representation, from which diagrams and matrices can be produced selectively and automatically. Also, the data dictionary is quite suitable for representing details that would be very confusing in a diagram or matrix, such as the Local Information-flow Models (LIMs) and their relationships with the GIM, the relationships between names in the GIM and in the LIMs, and details of database workload.

Some methodologies dispense with the GIM [NAVA82] and begin the design of the Conceptual Schema with a small number of applications, then add more applications, continually integrating the new applications with the old Conceptual Schema. This has the advantage of facilitating quick development of a prototype, but has the disadvantage of possible major revisions of the Conceptual Schema [JEFF82]. The safer procedure is to develop a GIM with careful control of detail, so that the level of effort is reasonable yet the GIM provides sufficient detail to guide the development of a relatively stable Conceptual Schema. This procedure is also likely to uncover important new interrelationships among LIMs, such as unexpected interrelationships among organizational components, and dependencies within them.

Note the similarity of the Local Information-flow Model and Global Information-flow Model development to Business Systems Planning (BSP) [MART82], which is also based on data flow. The primary difference, which is extremely important, is that each iteration of the Local Information-flow Model and Global Information-flow Model is followed by the development of the Conceptual Schema and External Schemas in the procedures described in this paper. This cyclical and iterative approach balances the data flow perspective with the data structure perspective, so that neither will be emphasized at the expense of the other. BSP, however, emphasizes the data flow perspective almost to the exclusion of the data structure perspective; high level data objects are identified, but their relationships and detailed structures must be developed by another methodology.

5.1 Information Used to Develop the GIM

Information that is relevant to the development of the GIM is obtained primarily from the previous iteration of the GIM and the newly developed LIMs. Other types of information are similar to those used to develop the LIM, except that they are at a higher organizational level.

1. The nature, objectives, and scope of the organization must be analyzed to ensure a compatible GIM.

2. The organizational perspective on decisions must be determined.

3. Organizational rules and policies must be analyzed.
4. Reports and forms must be examined.
5. Available resources must be determined.

5.2 Functions of the GIM

The primary function of the GIM is to guide the development of the Conceptual Schema. Other functions of the GIM are:

1. The GIM provides context for the development of the next iteration of the LIMs.
2. The GIM, like the LIMs, may assist in management planning to increase efficiency; the GIM provides a wider perspective on reducing data flow through changes in functions and organizational structures.
3. The GIM may also be used to design the interfaces among separate, loosely connected Conceptual Schemas, as may be appropriate among several large systems or a distributed database system.

5.3 Procedure for Developing the GIM

Figure 5 shows the four sequential steps in the development of the GIM. The steps are described in the following paragraphs.

GLOBAL INFORMATION-FLOW MODELING (GIM) PROCEDURE

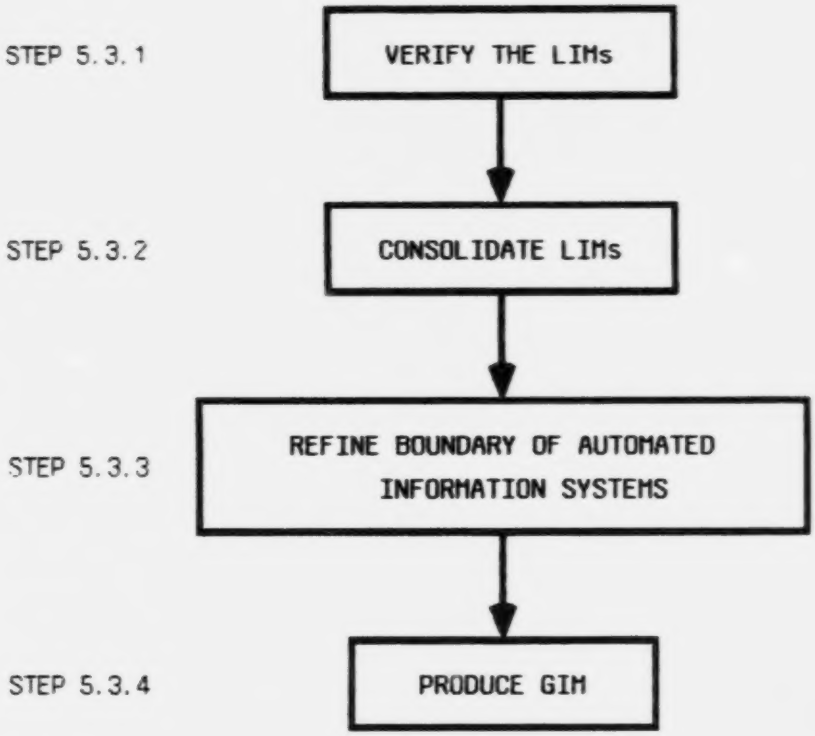

STEP 5.3.1
VERIFY THE LIMs
STEP 5.3.2
CONSOLIDATE LIMs
STEP 5.3.3
REFINE BOUNDARY OF AUTOMATED INFORMATION SYSTEMS
STEP 5.3.4
PRODUCE GIM

FIGURE 5

5.3.1 Verify the LIMs.

The LIMs are organized into a hierachy of organizational components, a separate but interrelated hierarchy of functions, and, possibly, a separate but interrelated hierachy of events. The function of this step is to verify that each new LIM is consistent with the objectives and constraints of the next higher level LIM in its hierarchy. Any inconsistencies require modification of either the lower-level LIM or the higher-level LIM. In the latter case, modifications may propagate all the way up the hierarchy and possibly affect the other hierarchies as well: such modifications may also propagate to the GIM, Conceptual Schema, and External Schemas. The following are the major considerations:

1. The data flow of a LIM must be consistent with that of its higher-level LIM. Each data object at the lower level should either appear at the higher level, or be a part of a higher-level data object, or have both source and sink within the lower-level LIMs. For example, assume that the higher level is a department, and the lower level consists of the branches within it. Data received by one branch from an outside source must be traceable to a departmental data source, but data sent to another branch might not appear at the departmental level.

2. Similarly, the data flow of the higher-level LIM must not be greater than the data flow of the LIMs that comprise it.

3. More generally, the scope of a lower-level LIM must be consistent with the scope of the higher-level LIM, where scope includes such non-data considerations as timing, resources, general objectives, and interrelationships with other hierarchies. For example, the branch should not have more time to perform a task than is available to the department, and should not perform functions that are not assigned to the department.

4. Similarly, the scope of the higher-level LIM must not be greater than the scope of the LIMs that comprise it.

5. If workloads have been developed, the workload of a LIM must be consistent with that of its higher-level LIM. Data volumes should be consistent. Each path through the lower-level data must either be entirely contained within the lower-level LIM or must be traceable to a path in the higher-level data. Priority, frequency, timing dependencies, and numbers of instances should be consistent.

6. Similarly, all of the paths in the higher-level LIM must appear in the lower-level LIM.

Step 5.3.1 Verify the LIMs

Function: To verify that each new LIM is consistent with the objectives and constraints of the next higher level

Output: LIMs organized in a hierarchy of organizational components, functions, or events

Team Members: User - AA, DA
Developer - AA, DA

Symbology: LIM diagram

Tools: Use DD to change entries and determine effects of change

Guidelines: Verify LIMs from top down

5.3.2 Consolidate LIMs.

The function of this step is to resolve synonyms that arise when different subsystems use different names for the same data flow and homonyms that arise when different subsystems use the same name for different data flows. Once detected, synonyms and homonyms are relatively easy to resolve. One of the synonyms is chosen for the GIM name, while the others are retained in the data dictionary as alternate names for the appropriate LIMs. For example, "part#" could be the preferred, global name, while "part-number" could be used within the context of a particular function, and be represented in the data dictionary as an alternate name. Only one object can be assigned the homonym for its GIM name; each of the other objects is assigned a new, unique name, and the homonym is assigned as an alternate name. For example, if "price" refers to both retail and wholesale price, then "price" could be used globally to refer to retail price, or locally within a particular function to refer to wholesale price; "wholesale-price" could be used to refer to wholesale price globally. Alternatively, "retail-price" and "wholesale-price" could be used globally, and "price" only locally.

Detection of synonyms is largely a manual process, but there are some clues that can be provided by the DDS or other computerized tool:

1. The primary means for detecting possible synonyms is data flow analysis, which can be performed by the DDS--for example, the DDS may be able to produce groups of data objects that have identical sources and sinks, which would indicate that the group members could be the same data object with different names in different subsystems.
2. Name analyses, such as keyword in context, are useful for suggesting possible synonyms.
3. Data element analysis may also help in suggesting possible synonyms by identifying data elements that have similar characteristics, such as their COBOL pictures or legal values.

Detection of homonyms should be primarily a process performed by the DDS--the DDS should reject any attempt to add conflicting characteristics to any data object. Situations in which two distinct objects have the same names and all other characteristics must be detected manually; however, if each object has a meaningful textual description, it is relatively simple to compare descriptions to determine whether they should be combined, or should be given separate names. Homonyms that are not resolved at this step may be resolved at a later step or later iteration of this step when more characteristics are known and therefore there is more likelihood of a conflict being detected by the DDS. Resolution at this step is a convenience but not a necessity.

Step 5.3.2 Consolidate LIMs

Function: Resolution of synonyms and honomyms

Output: One uniform model

Team Members: User - DA
Developer - DA

Symbology: Bubbles and lines

Tools: Use DD to store alternate names
Use name analyses such as keyword in context to detect synonyms

Guidelines: Standardize names in GIM
Use local synonyms whenever appropriate in LIMs

5.3.3 Refine Boundary of Automated Information System (AIS).

The function of this step is to refine the boundary of the automated information system that is being designed. This may reduce the scope of the logical database design and therefore reduce the effort expended in subsequent phases. Note that the final boundary will generally be three dimensional: organizational components, functions, and events. They must all be included in or excluded from the logical database design.

The criteria for drawing the boundary are primarily based on upper management goals as applied by the DA with possible technical advice from the DBA.

The boundary may be represented on a data flow diagram by a line, in a subsystem/data matrix by highlighting subsystems within the boundary or omitting subsystems outside the boundary, and in the data dictionary by a keyword or by relationships between a specific system and the subsystems within the boundary.

Step 5.3.3 Refine Boundary of Automated Information System (AIS)

Function: Reduce scope and refine the boundaries of the AIS

Output: Models of the AIS

Team Members: User - DA and upper level managers
Developer - DA and DBA

Symbology: Bubbles and lines

Tools: Use DD to represent specific system and subsystems within the boundary

Guidelines: Criteria for refining boundary are based on upper management goals

EXAMPLE OF A GLOBAL INFORMATION-FLOW MODEL

```
                            ____________                ____________
                           /            \              /            \
                          /Organizational\            /Organizational\
                          |  Component   |<---------  |  Component   |
                          |  Dependent   |            | Independent  |
                          \   on AIS     /            \    of AIS    /
                           \____________/              \____________/
                                 ^  |
                                 |  |
                                 |  |
                     |--------------------------------------------------
                     |           |  |
                     |           |  v
                     |
  ____________       |       ____________                ____________
 /            \      |      /            \              /            \
/Organizational\     |     /Organizational\            /Organizational\
|  Component   |---- |---->|  Component   |<---------  |  Component   |
|Providing Data|     |     | Interacting  |            | Interacting  |
\   to AIS     /     |     \   with AIS   /            \   with AIS   /
 \____________/      |      \____________/              \____________/
                     |             ^
                     |             |
                     |             |        Automated
                     |             |       Information
   Boundary-------->|             v         System
                     |                        (AIS)
                     |       ____________
                     |      /            \
                     |     /Organizational\
                     |     |  Component   |
                     |     | Interacting  |
                     |     \   with AIS   /
                     |      \____________/
                     |
                     |--------------------------------------------------
```

Figure 6

5.3.4 Produce GIM.

The function of this step is to provide additional quality assurance and documentation for the GIM. Use of a data dictionary is recommended. Details of how the data dictionary represents the GIM, what quality assurance reports are provided, and what documentation is to be produced must be determined by each organization to suit its own capabilities.

Step 5.3.4 Produce GIM

Function:	Provide final review and documentation for the GIM
Output:	Specification of components of GIM
Team Members:	User - DA and DBA Developer - DA and DBA
Symbology:	Bubbles and lines
Tools:	Use DD for corrections
Guidelines:	Quality assurance must be provided by application experts

6. CONCEPTUAL SCHEMA DESIGN

A Conceptual Schema (CS) is a description of the logical (hardware- and software-independent) structure of the data required by an organization. The phases concerned with development of the Local Information-flow Models (LIMs) and Global Information-flow Model (GIM) concentrated on the interactions between data and organizations, functions, or events; the structure and meaning of the data were not analyzed beyond the relatively simple resolution of synonyms and homonyms. This phase concentrates on the deep exploration of structure and meaning in terms of three important concepts: entity, relationship, and attribute. These concepts correspond very closely to the natural language constructs of noun, verb, and adjective. The following paragraphs, which define these concepts and provide brief examples, may be omitted by readers familiar with the Entity-Relationship-Attribute Model [CHEN80, CHEN81, CHEN82].

1. An entity is a type of real-world object or concept. For example, "employee," "project," and "position description" may be entities of interest to an organization. Note that only "employee" is a physical object--"project" and "position description" are both concepts. To appreciate the difference, consider that a "position description" may be recorded on a piece of paper. If the paper is copied or reproduced electronically in a database, the medium is changed, but the concept--the position description--is still the same. Therefore, the entity of interest is the message, not the medium.

2. A relationship is a type of association or correspondence among entities. For example, "works on" may be a relationship between "employee" and "project." An instance of a relationship is a fact or assertion--e.g., the phrase ´"12345" "works on" "design"´ could express the fact that the "employee" identified by the "employee number" "12345" is associated with the "project" entity identified by the "project-name" "design" through the relationship "works on." This example involves two entities and two instances of entities. A relationship may involve only one entity. For example, ´"design" "precedes" "implementation"´ is a relationship involving two instances of the entity "life-cycle phase." A relationship may also involve more than two entities-- e.g., ´"12345"

"works on" "design" "using" "Entity-Relationship-Attribute Approach"´ is an instance of a relationship ("works on" ... "using") among three entities ("employee," "project," and "technique").

3. An attribute is a property or characteristic which describes an entity or relationship. For example, the "employee" entity may have attributes such as "birth date," "marital status," and "annual salary," while the "works on" relationship may have attributes such as "hours per week," or "hours to date." Every entity must have an attribute or collection of attributes that distinguishes among entity instances (e.g., an "employee number" identifies a particular "employee"). A relationship may be without attributes, since each instance is identified by the entities that it associates (e.g., the relationship instance ´"design" "precedes" "implementation"´ is uniquely identified by "design" and "implementation," in that order).

6.1 Information Used to Develop the CS

Most of the information that is relevant to the development of the CS is provided indirectly by the GIM. Entities are the subjects of the data flows that were identified by the GIM, but they are generally not the data flows themselves. For example, a personnel report is not an entity unless there is system for tracking the production or distribution of the report, in which case each instance of the report might be identified by a control number. The subjects of the personnel report, e.g., "employee" and "project," would be entities.

6.2 Functions of the CS

The primary function of the CS is to provide a single logical structure for the database. Other functions include:

1. The CS provides input to the External Schema Design Phase.
2. The CS provides guidance in the choice of a data model (e.g., either a hierarchical, network, or relational data model may most easily represent the CS).
3. The CS provides guidance in the choice of a DBMS (e.g., a DBMS that easily represents the CS).
4. The CS provides guidance in the development and evaluation of the physical database design (the CS provides the definition of the logical data structure that the physical database must support).

The output of this phase may include the following:

1. For each entity of fundamental interest to the organization, its name, identifier (key), other attributes, synonyms, textual description, and relationships with other entities.
2. Entity-Relationship-Attribute diagrams [CHEN82].
3. Security, privacy, and integrity constraints.
4. Normalized relations [BEER79, BERN76, ZANI82].

6.3 Procedure for Developing the CS

Figure 7 shows the six steps in the development of the CS. The last step may reveal redundancies that will suggest repeating some or all of the preceding steps. The steps are described in the following paragraphs.

CONCEPTUAL SCHEMA (CS) DESIGN PROCEDURE

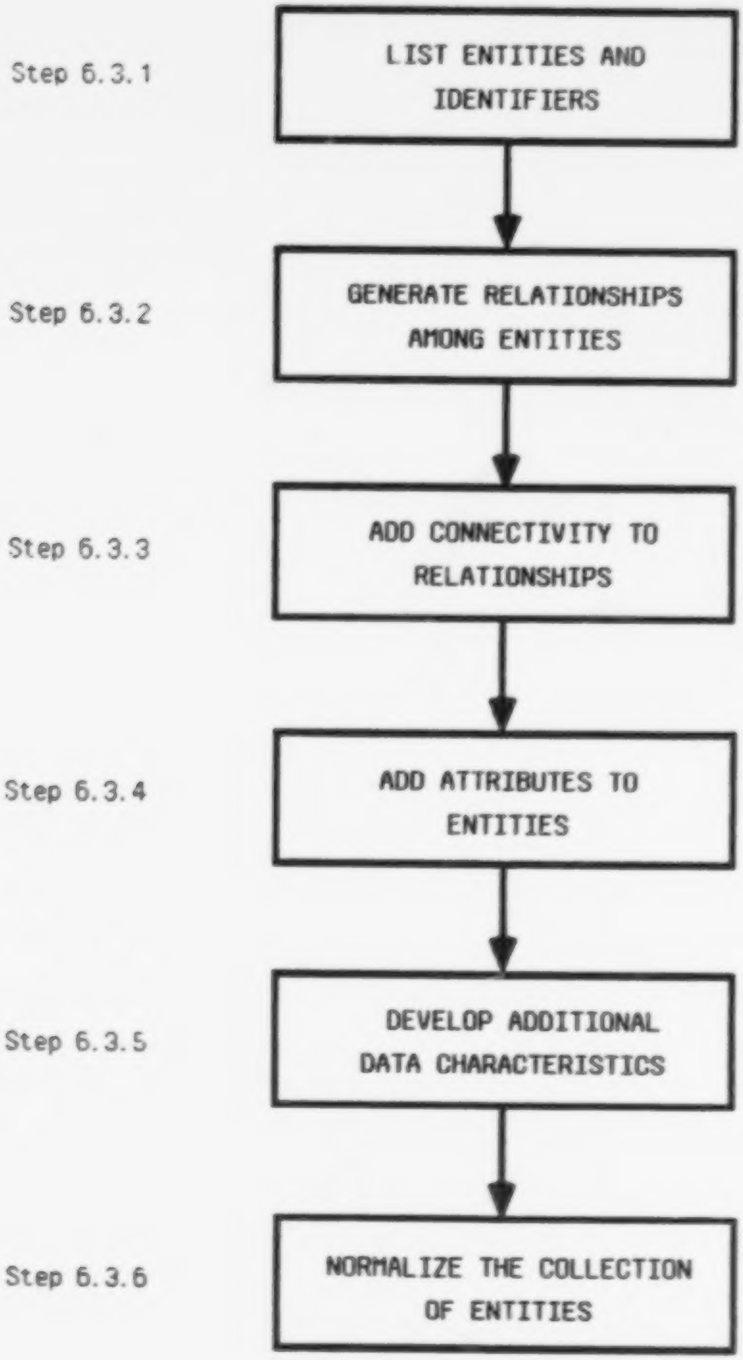

Step 6.3.1
LIST ENTITIES AND IDENTIFIERS
Step 6.3.2
GENERATE RELATIONSHIPS AMONG ENTITIES
Step 6.3.3
ADD CONNECTIVITY TO RELATIONSHIPS
Step 6.3.4
ADD ATTRIBUTES TO ENTITIES
Step 6.3.5
DEVELOP ADDITIONAL DATA CHARACTERISTICS
Step 6.3.6
NORMALIZE THE COLLECTION OF ENTITIES

FIGURE 7

6.3.1 List Entities and Identifiers.

The primary function of this step is to develop a list of entities that must be represented in the CS. Because of the inherent complexity of the real world that the CS models, this is considerably more difficult than one might assume. Some reasonable guidelines are presented below and discussed in the following paragraphs.

1. A data flow may suggest one or more entities.
2. An entity must have a meaningful name and description.
3. An entity must have an identifier.

In general, entities are the subjects of the GIM data flows; an entry in a report or form is usually an attribute which can identify or describe an entity. For example, an assignment matrix could have "project#" as the column heading, "employee-number" as the row title, and an "X" or blank as an indicator of assignment. The matrix itself is not an entity in most cases, but the "project#" and "employee-number" identify entities.

An entity should have a meaningful name consisting of a noun or noun phrase. If there is no obvious choice for the name of a proposed entity, then it is likely that it is not an entity. In addition, the entity must have an extended description that addresses topics such as the lifetime of an entity instance (e.g., is a "dependent" removed from the database when an "employee" resigns?) and criteria for inclusion (e.g., does "employee" include both hourly and salaried personnel?). For additional guidance, refer to [ATRE80, CHEN82, CURT82, KAHN79, ROUS81, SHEP76, SMIT78, SUST83, TEOR82].

An entity must have one or more identifiers (or keys). Each identifier is an attribute or combination of attributes which distinguishes among entity instances. For example, "employee-number," "project-name," and "PD#" could be the identifiers of "employee," "project," and "position description." The identifier of an entity may be composed of identifiers of other entities. For example, the identifier of "assignment" could be composed of the combination of the attributes "employee-number" and "project-name." Note that neither single attribute would uniquely identify a particular "assignment." Note also that "assignment" could equally well be identified by "SS#" and "project#," or even by a unique "assignment-number"--the important fact at this point is that an identifier can be found, so that "assignment" is a legitimate entity.

Analysis of the preceding example demonstrates that care must be exercised in finding an identifier and defining an entity:

- If the "employee" is released from the "project," is a record of the "assignment" retained?
- If so, how can such an assignment be distinguished from a current assignment?
- If the "employee" is returned to the "project," is the "assignment" still the same?

This analysis may indicate that the "employee-number" and "project-name" cannot constitute the identifier. Another attribute, such as "assignment-starting-date-and-time," may be needed for uniqueness. Another possibility is the "assignment-number;" the rules for handling multiple assignments could then be represented by the algorithm for determining the "assignment-number." For example, if the first "assignment-number" is 1, and each succeeding "assignment-number" is increased by 1, then multiple assignments of a given "employee" to a given "assignment" can always be distinguished.

Entities may be determined "top-down" by abstracting from the data flows and the GIM, or "bottom-up" by synthesizing from identifiers and their attributes [SHEP76]. The latter approach is greatly simplified by the use of a computer-based normalization program, as described in step 6.3.6. However, "top-down" is recommended because it forces the developer to concentrate on the semantic characteristics of the data; normalization can then be used to confirm the design.

Step 6.3.1 List entities and identifiers

Function:	Abstract data flows to determine entities
Output:	List of entities with descriptions and identifier
Team Members:	User - DA and DBA Developer - DA and DBA
Symbology:	Text
Tools:	Use DD to enter entities and identifier
Guidelines:	Be careful in defining an entity and finding the identifier for it Determine entities top-down

6.3.2 Generate Relationships among Entities.

The primary function of this step is to examine individual entities to see whether they can be subdivided into simpler, related entities, and to examine collections of entities to see whether they are related components of a more complex entity. A general guideline is to look at entities that share components. For example, "employee" and "assignment" share "employee-number;" obviously, there is a relationship between them. The data dictionary can be of great help in comparing entity structures.

The following are examples of common types of relationships [SUST83]:

1. Membership--a collection of similar secondary entities constitute another, primary, entity. The fiscal years in a five-year plan, the quarters in a fiscal year, or the cities in a state are examples of membership relationships. The relationship between the secondary and primary can be expressed by "in," "of," or "is a member of." The identifier of the primary may be required to identify each secondary; for example, a city name may be ambiguous unless the state is identified. The primary entity would

include properties common to all the secondary entities, while the secondary entities would have unique properties.

2. Aggregation--a collection of dissimilar secondary entities describes another, primary, entity. Generally all primary entities are related to similar collections of secondary entities. For example, each "employee" is described by the aggregation of "address," "salary-history," "education," etc., which are themselves entities. The relationship between the secondary and primary can be expressed by the phrase "is a property of" or "is a part of." The existence of a secondary entity is usually dependent on the existence of the primary entity.

3. Generalization--each of a collection of similar secondary entities can be considered to represent a special case of another, primary, entity. Different primary entities may be related to different types of secondary entities. For example, "salaried-employee" and "hourly-employee" are each roles of the primary entity "employee." The relationship between the secondary and primary can be expressed by the phrases "is a" or "is a type of." The existence of each secondary entity may be dependent on the existence of the primary entity; for example, every "salaried-employee" or "hourly-employee" must also be an "employee." The primary entity would include properties common to all the secondary entities, while the secondary entities would have unique properties.

These relationships correspond to the programming constructs of iteration (looping through the members of a collection), sequence (manipulating one after another of the aggregated properties), and selection (determining whether a particular role is played by the entity). All of these relationships can be developed bottom-up (from a given collection of secondary entities to the primary), to produce a simplified high-level structure, or top-down (from a primary to a collection of secondaries), to add more detail.

Another type of relationship which is occasionally useful is the following:

4. Precedence--the existence of one entity in the database must precede the existence of another entity in the database. For example, a "proposed-budget" must precede an "approved-budget;" once an "approved-budget" has been entered, however, its existence is independent of the "proposed-budget."

Other, more specialized relationships are discussed in [SUST83].

Diagrams are recommended as a convenient way of communicating with the application specialists. Examples are given below.

EXAMPLE OF AN ENTITY-RELATIONSHIP DIAGRAM

```
        -------------
       |             |
       |     E1      |
       |             |
       |             |
        -------------
              ^
              |
              |
              |  Relationship name
              |
              |
              v
              v
        -------------
       |             |
       |     E2      |
       |             |
       |             |
        -------------
```

Figure 8

This example states that entity "E1" has a relationship with another entity "E2." The single and double arrows indicate that an instance of "E1" may be associated with many instances of "E2," while each instance of "E2" is associated with one instance of "E1."

ALTERNATE NOTATION FOR AN ENTITY-RELATIONSHIP DIAGRAM

```
 -------------
|             |
|     E1      |
|             |
|             |
 -------------
       ^
       |
      / \
     /   \
    / Rel \
    \Name /
     \   /
      \ /
       |
       v
       v
 -------------
|             |
|     E2      |
|             |
|             |
 -------------
```

Figure 9

The alternate notation is somewhat more cumbersome but it does have the advantage of emphasizing the importance of relationships, and is readily extended to include relationships among more than two entities and relationships with attributes.

In general, the simplicity of labeled lines is preferred. A relationship among more than two entities should usually be transformed into an entity which has simple relationships with those entities. For example,

REPLACING A RELATIONSHIP WITH AN ENTITY

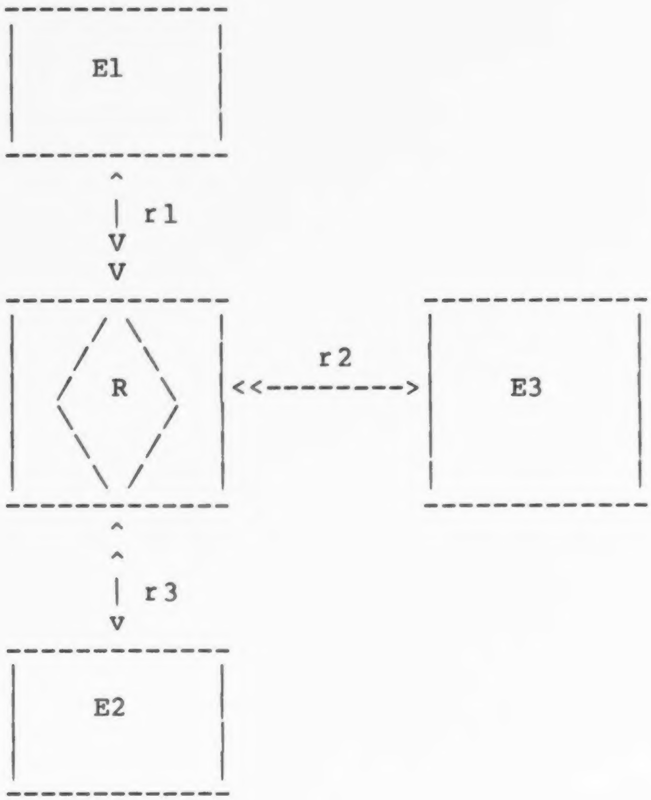


Figure 10

The complex relationship R has been replaced by an entity; the diamond within the rectangle indicates that R may be an entity on one diagram and a relationship on a less detailed diagram. New relationships, r1, r2, and r3 must be added unless they are obvious. The fact that an "employee" uses a particular "skill" on a particular "project" would be represented by such a diagram; E1, E2, and E3 would represent "employee," "skill," and "project," while R could be a relationship or an entity identified by the "employee," "skill," and "project" identifiers.

Step 6.3.2 Generate relationships among entities

Function: Revise entities

Output: Entities and relationships

Team Members: User - DA and DBA
Developer - DA and DBA

Symbology: Entity-Relationship diagrams

Tools: Add relationships to DD

Guidelines: Look for common types of relationships

6.3.3 Add Connectivity to Relationships.

The primary function of this step is to suggest new entities or ways in which entities can be combined. A secondary function is to provide quantitative data useful to physical database design.

Connectivity describes a relationship between two entities-- how many instances of one entity are associated with how many instances of the other entity. For example, if an "employee" can have only one "manager," but a "manager" can manage many employees, then the relationship "manages" is "1 to many." If a reasonably good number can be given for the "many," that may assist in physical database design. However, the most important situations for logical database design are the following:

- Most relationships will have connectivity "1 to many" or "many to 1."
- If the connectivity is "1 to 1," then the two entities should be combined, provided that the result can be given a meaningful name and description. For example, if a "project" always has exactly one "manager," and a "manager" always has exactly one "project," then the two entities can be combined.

(Note the use of the word "always." In the real world it is likely that there will be periods of transition when a "manager" has no "project," or more than one "project," or a "project" has no "manager." In reality, then, the connectivity might be "0,1 to 0,1,2," and the entities should not be combined.)

- If the connectivity is "1 to 0,1" then this often indicates generalization. For example, the relationship between "employee" and "salaried-employee" is "1 to 0,1," since the "employee" could be an "hourly-employee." The "salaried-employee" entity cannot exist unless the "employee" entity exists.

- If the connectivity is "many to many" (or numbers indicating a similar situation), then the relationship should be replaced by an entity. For example, if there is a "many to many" relationship between "employee" and "manager" (i.e., matrix management), then a new entity, such as "assignment of employee to manager" should be created, and the "many to many" relationship replaced by two "1 to many" relationships. This leads to more entities but simplifies relationships and also simplifies the mapping of the logical database design into a conventional data model.

An example of a diagram with connectivity is shown below.

EXAMPLE OF AN ENTITY-RELATIONSHIP DIAGRAM WITH CONNECTIVITY

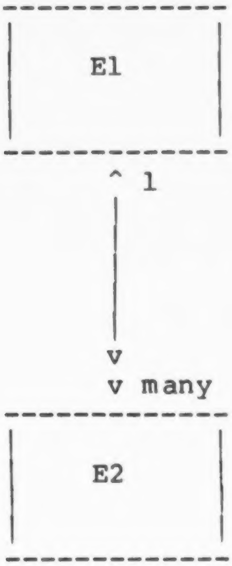


Figure 11

Step 6.3.3 Add connectivity to relationships

Function:	Determine connectivity and provide quantitative data to physical database design
Output:	Annotated relationships
Team Members:	User - DA and DBA Developer - DA and DBA
Symbology:	Extended E-R diagrams
Tools:	Add connectivity information to DD
Guidelines:	Eliminate 1 to 1 and many to many relationships

6.3.4 Add Attributes to Entities.

The primary function of this step is to add detail to the entity descriptions in the data dictionary and diagrams. Two strategies are possible:

1. If there is a collection of known attributes (e.g., data elements), then this step can be performed "bottom-up." Each attribute is assigned to an entity (or entities) which identifies a unique instance of that attribute. If no entity is appropriate, one is created, relationships are developed, and so on.
2. This step can be performed "top-down" by examining each entity to determine appropriate descriptors. This procedure is recommended during high-level iterations, when attributes are data collections rather than data elements.

The attributes are represented in the data dictionary by being "contained in" an entity [FIPS80], and in the diagrams by some notation such as that in the following example, where "A1" is the attribute:

EXAMPLE OF AN ENTITY-RELATIONSHIP-ATTRIBUTE DIAGRAM

```
 -------------                   ---
|             |                 /   \
|     E1      |----------------|  A1 |
|             |                 \   /
|             |                  ---
 -------------
      ^ 1
      |
      |
      | S
      |
      |
      v 0,1
 -------------
|             |
|     E2      |
|             |
|             |
 -------------
```

Figure 12

The relationship S could be an agreed-upon symbol to indicate that E2 is a subtype of the entity El.

Another function of this step is to simplify the CS by eliminating unnecessary entities. The rule for doing this is very simple:

o If an entity is single-valued in every relationship with other entities, then it can be eliminated by moving its attributes (including the identifier) into those entities.

For example, suppose that "hourly-pay-scale" is an entity with the attribute and identifier "dollar-amount," and its only relationships are "many to 1" from "salaried-employee" and "hourly-employee" to "hourly-pay-scale." Then "dollar-amount" should be assigned to "salaried-employee" and "hourly-employee," and "hourly-pay-scale" should be eliminated. The justification is simple: "dollar-amount" is single-valued in every relationship, so it acts like a descriptor--i.e., an attribute.

Step 6.3.4 Add attributes to entities

Function:	Add attributes to the entity descriptions
Output:	E-R-A diagrams
Team Members:	User - DA and DBA Developer - DA and DBA
Symbology:	E-R-A diagrams
Tools:	Add attributes to DD
Guidelines:	Simplify by eliminating unnecessary entities

6.3.5 Develop Additional Data Characteristics.

The function of this step is to add additional constraints, such as security and integrity, to the entity and relationship descriptions in the data dictionary. These constraints are important but are not easily represented on a diagram; the recommendation is to keep the diagrams simple by representing these constraints only in the data dictionary.

Step 6.3.5 Develop additional data characteristics

Function: Add security, integrity, and other constraints

Output: E-R-A diagrams and updated DD with detailed description of data

Team Members: User - DA and DBA
Developer - DA and DBA

Symbology: E-R-A diagrams

Tools: Add constraints to DD

Guidelines: Keep the diagrams simple

6.3.6 Normalize the Collection .

The primary function of this step is to ensure that the collection of entities is optimal in the following sense:

1. Each non-key attribute is identified only by the simplest possible identifiers. For example, "supplier-address" should not be in a "supplier-part" entity (identified by the combination of "supplier-name" and "part-number") if "supplier-address" is uniquely identified by "supplier-name" alone.

2. Redundant non-key attributes are eliminated. For example, if the "branch" entity contains "division#" and "department#," and the "division" entity (identified by "division#") also contains "department#," then "department#" can be eliminated from "branch." The "department#" can be determined from the unique "division" entity identified in the "branch," so "department#" is redundant in "branch."

3. Entities with the same identifier are combined.

4. Entities with equivalent identifiers (identifiers that identify each other) are combined.

The first two conditions, plus the condition that attributes are single-valued (which was required in step 6.3.4), are sufficient to ensure that the entities are in Third Normal Form [BERN76]. The third and fourth conditions ensure that the entities are in the more rigorous Elementary Key Normal Form (EKNF) [ZANI82], which minimizes the total number of entities. A computer algorithm to obtain EKNF is described in [BEER79, BERN76]; the proofs of correctness and minimality are complex, but the algorithm itself is quite simple.

Commercially available programs perform various levels of normalization [MART77]. A good program should interface to a data dictionary to obtain identifiers and the attributes that they identify, and should provide EKNF as well as various reports, traces, and diagrams. The objective of the preceding steps of this phase is to do such a good job of identifier analysis that the normalization program will produce exactly the entities that are input to it. Experience indicates that discrepancies between the input and output entities are often caused by more serious and subtle errors than those found by the normalization program; the program exposes errors, but its "corrections" are sometimes difficult to understand, and should not be accepted without thorough analysis. A normalization program should definitely not be used as a substitute for careful thought.

Step 6.3.6 Normalize the collection of entities

Function: Remove redundancies and detect errors

Output: Normalized entities

Team Members: User - System analyst and DBA
Developer - DA and DBA

Tools: Normalization program

Guidelines: Careful manual analysis as well as use of the automated tools

---

7. EXTERNAL SCHEMA MODELING

An External Schema (ES) is a subschema (part) of a Conceptual Schema (CS) that is relevant to a Local Information-flow Model (LIM). A LIM, in turn, represents the information requirements of a user, group of users, application program, or application system. An ES includes all entities, relationships, and attributes needed by the LIM. Local names are possible--for example, the Conceptual Schema may have an entity called "employee-number" which is "emp-no" in the personnel ES. An ES reflects the way information is used by an individual task or decision.

7.1 Information Used to Develop the ES

The primary sources of information needed to develop an ES are the CS and the relevant LIM as represented in the data dictionary. If the LIM is inadequate in scope or detail, then it should be expanded using additional information from the sources listed in section 4.1.

7.2 Functions of the ES

The primary function of an ES is to help users and programmers interact with the database by presenting a simplified view of the database in terms which are familiar to them. An ES has the following secondary functions:

1. Detailed iterations of the ES provide one of the inputs to physical database design--they describe the workload, originally developed in terms of LIMs, in terms of the CS.
2. An ES is a piece of the CS which can be assigned privacy and security locks during physical database design and implementation phases.
3. An ES provides quality control of the CS--if the ES cannot be constructed from the CS, then the CS is incomplete. Also, if there are portions of the CS which are not required by any ES, then those portions

are unnecessary or are information sources that are not being utilized by any LIMs. During the early iterations of the logical database design process the ESs will be useful only for comparing high-level descriptions of very general categories of data (e.g., data needed for the support of management decisions), since the relevant LIMs will be based on an organizational perspective and will not have much detail. In addition, the LIMs may not indicate what information is to be in the database and what is to be provided by some other source. During later iterations, the ESs will provide a much more accurate means for ensuring CS quality.

7.3 Procedure for Developing the ES

Figure 13 shows the three sequential steps in the development of the ES. The steps are described in the following paragraphs.

EXTERNAL SCHEMA (ES) MODELING PROCEDURE

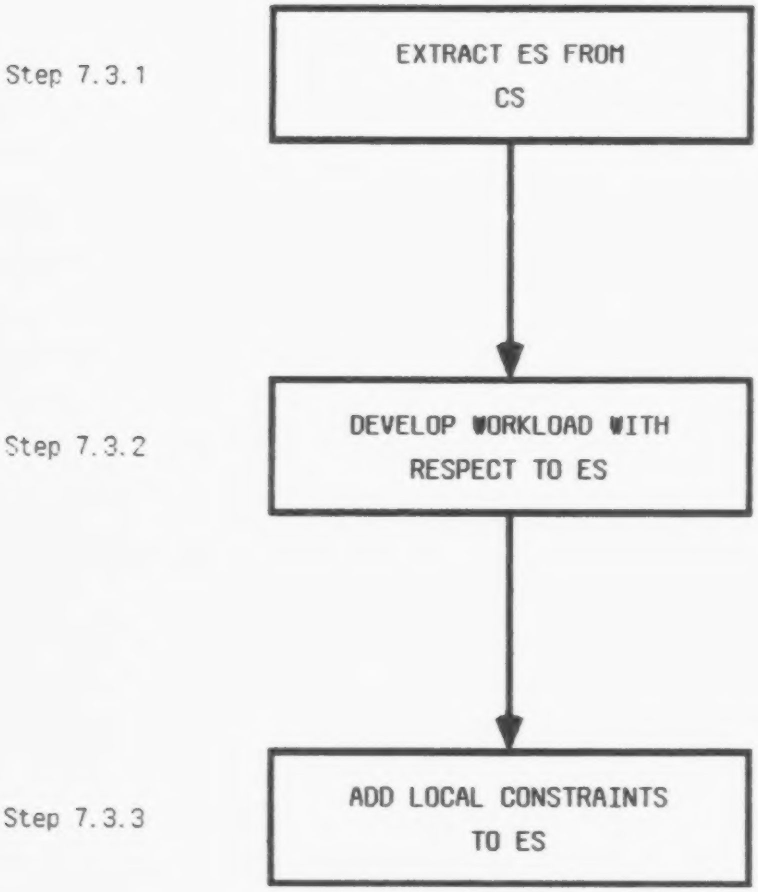

Step 7.3.1
EXTRACT ES FROM CS
Step 7.3.2
DEVELOP WORKLOAD WITH RESPECT TO ES
Step 7.3.3
ADD LOCAL CONSTRAINTS TO ES

FIGURE 13

7.3.1 Extract an ES from the CS.

The primary function of this step is to decide what parts of the CS are required by a particular LIM. First, data flows must be classified into those requiring data from the database and those that are independent of the database [JEFF82]. The data collection may be obtained from or stored in a private file or other non-database location if any of the following are true:

1. The data collection is of interest to only a single user or application and therefore need not be shared.
2. The data collection is transitory, as in a temporary working file, and would not exist long enough to be relevant to other users or applications.
3. The data collection is incomplete or inconsistent, as in a partially completed update, or consists only of references or keys to other data, as in a file of references to data of particular interest to decision support.

In general, a data collection should be obtained from or stored into the database if all of the following are true:

1. The data collection is of interest to many users or applications and should therefore be shared.
2. The data collection is sufficiently long-lived to have many uses.
3. The data collection represents a consistent, complete view of the real world.

There are then two situations that can be distinguished:

- This LIM is not a part of any LIM for which an ES has already been constructed. For example, this LIM might be a top-level organization, function, or event. In this case, the ES will consist of high-level entities, relationships, and attributes from the CS. If a Data Dictionary System (DDS) is available, it should be employed to extract only high-level data objects. These objects will then be manually compared with the data flows of the LIM to determine what parts of the CS are needed by the LIM.

- Alternatively, this LIM is a part of a higher-level LIM for which an ES has already been constructed. For example, this LIM may be a part of a function for which there is an ES. In this case, the ES is based on the higher-level ES. The DDS should be used to extract the data objects relevant to the higher-level ES, and the lower-level data objects which are contained within them. The resulting collection of data objects must then be compared with the data flows of the LIM to verify that all data required by the LIM is in the higher-level ES, or is a part of some data object in the higher-level ES (the DDS can greatly reduce the effort involved in this comparison). If not, the higher-level ES must be extended to include the missing data. The lower-level ES will then consist of the relevant parts of the higher-level ES plus additional entities, relationships, and attributes required by the more detailed level of analysis.

The final result of this step is a diagram of selected parts of the CS plus additional entries in the data dictionary to relate the selected data to the LIM.

Step 7.3.1 Extract an ES from the CS

Function: Decompose CS based upon the particular LIM

Output: Decomposed E-R-A diagram

Team Members: User - Programmers, analysts, and DBA
Developer - DA and DBA

Symbology: E-R-A diagrams

Tools: Use DD to relate data to LIM

Guidelines: Verify the extracted ES with LIM

7.3.2 Develop Workload With Respect to ESs.

The primary function of this step is to translate the workload, originally developed in terms of data flow in the LIM, into data access and update in the ES. The preceding step determined what parts of the database, if any, are required for each data flow, while step 4.3.5 determined the frequency, sequence, and selectivity with which each function uses and updates data. Therefore, this step involves two alternatives for each data collection in the LIM workload sequence:

- If the data collection is not database data, then nothing need be done.
- If the data collection is database data, then an appropriate access path must be determined. That is, given the data available at that point in the sequence, what entities and relationships must be accessed to arrive at the required entities? If a path cannot be found, there is an error, which must be corrected by modifying the LIM (e.g., by revising the workload), modifying the partially completed ES (e.g., by changing the distribution of database and non-database data), or modifying the CS (e.g., by adding a new relationship). If a path can be found, it is added into the workload sequence for the ES.

The resulting database workload should be represented in the data dictionary by a sequence of programs or modules interacting with the database objects. Three kinds of interactions with entities must be represented:

- Data use--an entity instance is accessed because various attributes are needed for some computation, report, or control purpose.
- Data update--an entity instance is added or modified.
- Data access--an entity instance is part of a path but has no directly relevant attributes. The entity might be removed from the path, with an improvement in database performance, if the Internal Schema has an appropriate relationship to bypass the entity.

As noted in step number 4.3.5, there are two types of interactions with attributes:

- Entity retrieval--an attribute is needed to determine whether an entity instance is needed by the function.
- Attribute selection--an attribute instance is required for a computation, report, control, or update purpose.

There is one type of interaction with relationships:

- Path component--the relationship is part of a path. Note that the direction is important.

The paths may also be represented graphically by an overlay on an ES or CS diagram [MART84, MCCL84, SUST84]. This provides a simple representation that can be easily understood and verified by application specialists, but is not a substitute for the data dictionary.

Step 7.3.2 Develop workload with respect to ES

Function: Specifications for physical design

Output: Workload specifications

Team Members: User - Programmers, analysts, and DBA
Developer - Analysts, DA and DBA

Symbology: E-R-A diagram with path overlay

Tools: Update DD to add workload information

Guidelines: Identify access path to avoid errors

7.3.3 Add Local Constraints to the ES.

The purpose of this step is to add any unique constraints imposed on or by the LIM. Examples of such constraints include security and privacy restrictions, local rules for edit and validation, and local integrity constraints.

Step 7.3.3 Add local constraints to the ES

Function: Add local constraints to each ES

Output: Updated E-R-A diagrams and updated DD

Team Members: User - Programmers, analysts, and DBA
Developer - DA and DBA

Symbology: E-R-A diagrams

Tools: Update DD to add constraints

Guidelines: Identify unique constraints imposed on or by the LIM

8. CONCLUSIONS

This report presents a Logical Database Design methodology with the following characteristics:

- There are four phases: Local Information-flow Modeling, Global Information-flow Modeling, Conceptual Schema Design, and External Schema Modeling.
- The phases are executed iteratively to control complexity and to provide a means for verifying the results of the different phases against one another.
- Analysis is performed from different points of view (organization, function, and event) in order to ensure that the logical database design accurately reflects all reasonable information requirements of the organization.
- The methodology recommends computer support from a Data Dictionary System, in order to conveniently and accurately handle the volume and complexity of design documentation and analysis, and to provide ready access to work already accomplished.
- Logical database design is integrated into the complete system life cycle.

The purpose of this methodology is to assist in the design of very large and complex information systems, where the effects of poor logical database structures can result in expensive, time-consuming system development efforts whose end results are ineffective and inefficient. The methodology emphasizes both the need for speed, so that the design will be completed in time to be useful, and the need for quality control, to ensure that the design is consistent, complete, and satisfies the eventual users.

9. ACKNOWLEDGMENTS

We wish to acknowledge the contributions of Daniel Benigni, Joseph Collica, Mark Skall, and T.C. Ting for their work on the outline for this report; Peter Chen, Ilchoo Chung, and Dennis Perry for their research reported in [CHEN82]; Nick Roussopoulos and Raymond Yeh for their research reported in [ROUS81]; Bernard Thomson of the Navy Program Planning Office (OP-901M), for providing an early test of the methodology; and Harold Stout of the Command Information Systems Office, Military Sealift Command, for his support and extensive testing of the methodology. Special thanks are due to Kang Cheng, for her excellent diagrams.

10. REFERENCES AND SELECTED READINGS

[AFIF84] Afif, A., "Automated Enterprise Modeling and Database Design," *Proceedings Trends and Applications 1984: Making Database Work*, IEEE Computer Society Press, 1984, pp. 247-256.

[ANSI84] American National Standards Institute (ANSI) Technical Committee X3H4, *Working Draft American National Standard IRDS: Part 1: Core Standard*, dated December 1984.

[ATRE80] Atre, S., *Data Base: Structured Techniques for Design, Performance, and Management*, John Wiley and Sons, Inc., 1980.

[BEER79] Beeri, Catriel and Bernstein Philip A., "Computational Problems Related to the Design of Normal Form Relational Schemas," *ACM Transactions on Database Systems*, Vol. 4, No. 1, March 1979, pp. 30-59.

[BERN76] Bernstein, Philip A., "Synthesizing Third Normal Form Relations from Functional Dependencies," *ACM Transactions on Database Systems*, Vol. 1, No. 4, December 1976, pp. 277-298.

[CARL80] Carlis, John V., "An investigation into the Modeling and Design of Large, Logically Complex, Multi-user Databases," Ph. D. thesis submitted to University of Minnesota, Minneapolis, Minnesota 55455, December 1980.

[CARL81] Carlis, John V. and March, Salvatore T., "A Multiple Level Descriptive Model for Expressing Logical Database Design Problems and Their Physical Solutions," Working Paper Series MISRC-WP-81-10, University of Minnesota, Minneapolis, Minnesota 55455, March 1981.

[CERI83] Ceri, Stefano, (ed.), *Methodology and Tools for Database Design*, North-Holland Publishing Company, 1983.

[CHEN80] Chen, P. P., (ed.), *Proceedings of 1st International Conference on Entity-Relationship Approach to Systems Analysis and Design*, North-Holland

Publishing Company, May 1980.

[CHEN81] Chen, P. P. (ed.), *Entity-Relationship Approach to Information Modeling and Analysis*, ER Institute, P.O. Box 617, Saugus, CA 91350, October 1981.

[CHEN82] Chen, P. P., Chung, Ilchoo, and Perry, Dennis, "A Logical Database Design Framework," NBS-GCR-82-390, NTIS No. PB82-203316, May 1982.

[CURT82] Curtice, Robert M. and Jones, Paul E., *Logical Database Design*, Van Nostrand Reinhold Company, 1982.

[DEMA78] DeMarco, Tom, *Structured Analysis and System Specification*, Yourdon Inc., 1133 Avenue of the Americas, New York, NY 10036, 1978.

[FIPS80] Federal Information Processing Standards (FIPS), *Guideline for Planning and Using a Data Dictionary System*, FIPS Publication 76, U.S. Department of Commerce, National Bureau of Standards, August 1980.

[GALL84] Gallagher, L. J. and Draper, J. M., *Guide on Data Models in the Selection and Use of Database Management Systems*, NBS Special Publication 500-108, National Bureau of Standards, January 1984.

[GANE79] Gane, Chris and Sarson, Trish, *Structured Systems Analysis : Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1979.

[JACK83] Jackson, M. A., *System Development*, Prentice-Hall International, 1983.

[JEFF82] Jefferson, David K., *Information Systems Design Methodology: Overview*, DTNSRDC-82/043, David W. Taylor Naval Ship Research and Development Center, Bethesda, MD 20084, NTIS No. ADA-115902, May 1982.

[KAHN79] Kahn, B. K., "A Structured Logical Database Design Methodology," Ph. D. thesis submitted to the University of Michigan, Ann Arbor, Michigan 48109, 1979.

[KONI81] Konig, P. A. and Newton, J. J., *Federal Requirements for a Federal Information Processing Standard Data Dictionary System*, NBSIR 81-2354, U.S. Department of Commerce, National Bureau of Standards, September 1981.

[LUMV79] Lum, V.Y., et al., "1978 New Orleans Data Base Design Workshop Report," Proc. 5th International Conference on Very Large Databases, The Institute of Electrical and Electronics Engineers, Inc. (IEEE), October 1979.

[MACD82] MacDonald, I. G. and Palmer, I. R., "System Development in a Shared Data Environment - The D2S2 Methodology," in Information Systems Design Methodologies,: A Comparative Review, Olle, T. W. et al (eds.) North-Holland Publishing Company, 1982.

[MARC78] March, Salvatore T., Jr., "Models of Storage Structures and the Design of Database Records Based Upon a User Characterization," Ph. D. thesis submitted to Cornell University, May 1978.

[MARC84] March, S. T., Ridjanovic, D. and Prietula, M., "On the Effects of Normalization on the Quality of Relational Database Designs or Being Normal is Not Enough," Proceedings Trends and Applications 1984: Making Database Work, IEEE Computer Society Press, 1984, pp. 257-261.

[MART77] Martin, James, Computer Database Organization, Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1977.

[MART82] Martin, James, Strategic Data-Planning Methodologies, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[MART84] Martin, James, "The Spring 1984 James Martin Seminar Documentation," Volumes I, II, and III, Technology Transfer Institute, 741 10th Street, Santa Monica, CA 90402, 1984.

[MCCL84] McClure, Carma L. "Structured Techniques for Fourth Generation Languages," Technology Transfer Institute, 741 10th Street, Santa Monica , CA 90402, 1984.

[MYER78] Myers, Glenford J., Composite/Structured Design, Van Nostrand Reinhold Company, 1978.

[NAVA82] Navathe, S. B. and Gadgil, S. G., "A Methodology for View Integration in Logical Database Design," in Proceddings of the Eighth International Conference on Very Large Databases, Mexico City, September 1982.

[ORRK82] Orr, Ken and Associates, Inc., Data Structured Systems Development Methodology, Ken Orr and Associates, Inc., 1725 Gage Blvd., Topeka, KS 66604. 1982.

[ROSS77] Ross, D. T. and Schoman, K. E., Jr., "Structured Analysis for Requirements Definition," IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, PP. 6-15, 1977.

[ROUS81] Roussopoulos, N. and Yeh, R. T., "Database Logical Schema Design," NBS GCR 82-411, NTIS No. PB 83-195743, 1981.

[SAKA83] Sakai, H. "Entity-Relationship Approach to Logical Database Design," in Davis, C. G. et al (eds.) Entity-Relationship Approach to Software Engineering, North-Holland, 1983.

[SHEP76] Sheppard, D. L., "Database Methodology -- Parts I and II," Portfolios 23-01-01 and 23-01-02, Design and Development, Database Design, Auerbach Publishers, 1976.

[SMIT78] Smith, J. M., and Smith, D. C., "Principles of Database Conceptual Design," NYU Symposium on Database Design, May 1978.

[SOFT79] SofTech, Inc., "IDEF - Architect's Manual," Material Supplied by Project 112 Task 2 Coalition, Consisting of Hughes Aircraft Company and Northrop Corporation. Manual Prepared by SofTech, Inc., 460 Totten Pond Road, Waltham, MA 02154, August 1979.

[SUND78] Sundgren, B., "Database Design in Theory and Practice: Towards An Integrated Methodology," Proceedings of 4th International Conference on Very Large Databases, The Institute of Electrical and Electronics Engineers, Inc. (IEEE), 1978.

[SUST83] Su, Stanley Y. W., "SAM*: A Semantic Association Model for Corporate and Scientific-Statistical Databases," Information Sciences, Vol. 29, 1983, PP. 151-199.

[SUST84] Su, Stanley Y. W., "Processing Requirement Modeling and Its Applications in Logical Database Design," in Yao, B. S. (ed.) Principles of Database Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, to be published in 1984.

[TEOR82] Teorey, T. J., and Fry, J. P., Design of Database Structures, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1982.

[ZANI82] Zaniolo, Carlo, "A New Normal Form for the Design of Relational Database Schemata," ACM Transactions on Database Systems, Vol. 7, No. 3, September 1982, PP. 489-499.

APPENDIX A

Agency Financial Management System

INTRODUCTION

A Federal agency is designing a financial management system. None of the applications systems offered by software vendors seem to gracefully accommodate the agency's code structure and its cost accounting procedures for its reimbursable divisions. As a matter of fact, although the individuals on the team surveying these packages are each expert in a particular subject area, they lack a good overview of what their agency's requirements are, or should be.

A primary objective of the design effort is to gain an organizational perspective of the agency's financial data. The logical database design can then be used to develop a system (either in-house or on contract), purchase a system (once requirements are understood) or specify modifications which would be needed if a system were purchased from a vendor or obtained from another agency.

An important consideration in the logical database design project is that the agency's appropriation from Congress constitutes only 63% of the operating budget. Additional income is provided by contracts with other government agencies and the sale of goods and services to the public sector. The financial management system must be able to charge back costs to customers. Another important consideration is that there is an existing payroll system which must interface with the financial management system.

An example of a reimbursable division is Instrument Fabrication Division, IFD, whose income from services to other government agencies represents 8% of the agency's budget. IFD relies on other divisions within the agency for functions such as procurement and accounting. IFD finances all management and support services by applying a fixed-rate surcharge to the labor base in some of its own units.

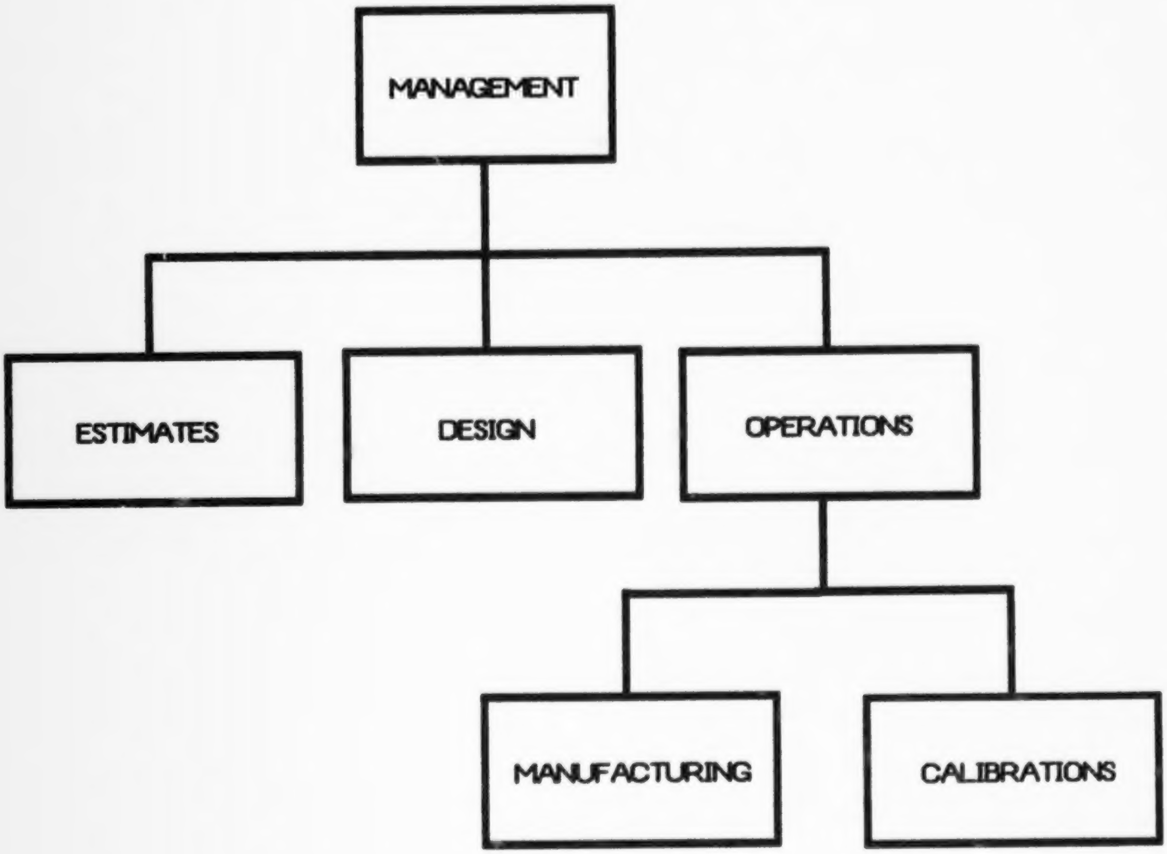
The following examples are intended to show some of the types of documentation which are gathered or produced in a logical database design.

These examples have been simplified so that the amount of detail does not obscure the intent of the example. However, in some instances enough detail is left in so that the reader may appreciate the sheer volume of the items of information to be gathered, analyzed and organized in logical database design. The result is, unfortunately, an uneven level of detail.

Even the sample system chosen, "Agency Financial Management System," is limited in scope, showing some aspects of normal in-house financial management for a service-oriented agency. Other federal agencies, whose mission is to administer or disburse government funds, would consider this example system a minor subsystem. In general, logical database design for financial management should consider the unique mission of the agency and the extent to which financial data can be used to support that mission.

INSTRUMENT FABRICATION DIVISION

Organizational Chart

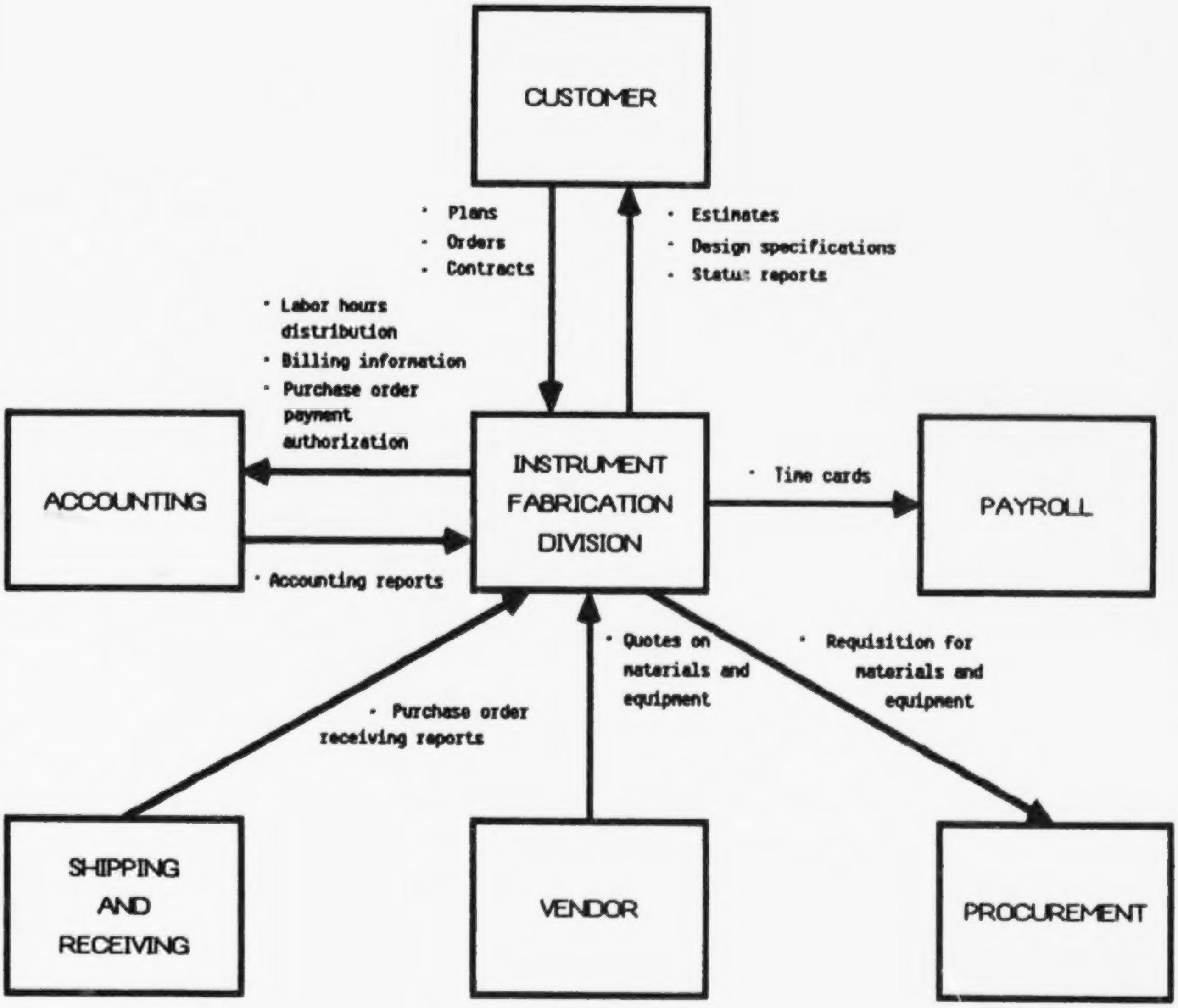

MANAGEMENT
ESTIMATES
DESIGN
OPERATIONS
MANUFACTURING
CALIBRATIONS

MISSION

The mission of Instrument Fabrication Division is to design and manufacture high-precision, one-of-a kind instruments in support of the agency's scientific research divisions. This service is available to other government agencies as well as the public. All instruments are manufactured on a reimbursable basis.

INSTRUMENT FABRICATION DIVISION

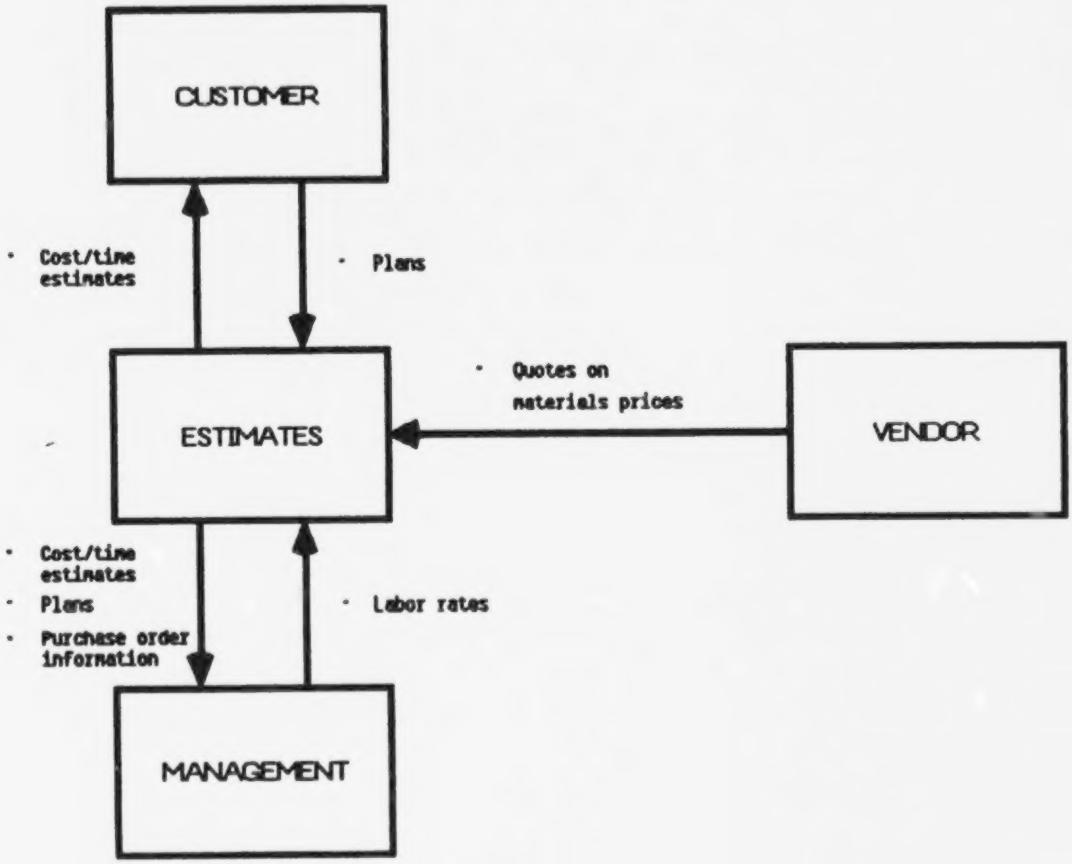
High Level Local Information-flow Model


CUSTOMER
· Plans
· Orders
· Contracts
· Estimates
· Design specifications
· Status reports
· Labor hours distribution
· Billing information
· Purchase order payment authorization
ACCOUNTING
INSTRUMENT FABRICATION DIVISION
· Time cards
PAYROLL
· Accounting reports
· Quotes on materials and equipment
· Requisition for materials and equipment
· Purchase order receiving reports
SHIPPING AND RECEIVING
VENDOR
PROCUREMENT

INSTRUMENT FABRICATION DIVISION

Local Information-flow Model

ESTIMATES Unit

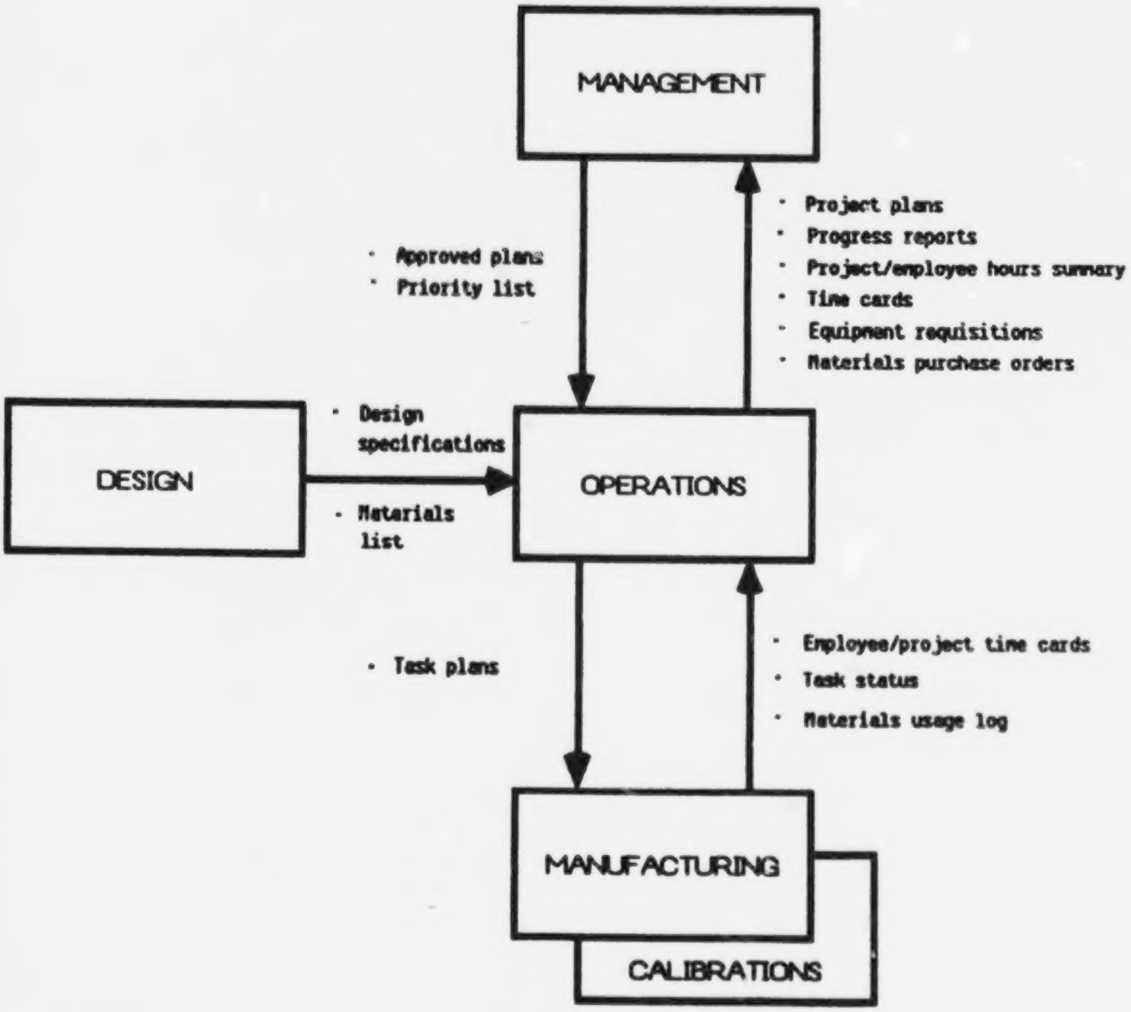

CUSTOMER
• Cost/time estimates
• Plans
ESTIMATES
• Quotes on materials prices
VENDOR
• Cost/time estimates
• Plans
• Purchase order information
• Labor rates
MANAGEMENT

NOTES

Estimates are free to customers. The ESTIMATES unit is not reimbursed directly for services.

INSTRUMENT FABRICATION DIVISION

Local Information-flow Model
OPERATIONS Unit


MANAGEMENT
• Approved plans
• Priority list
• Project plans
• Progress reports
• Project/employee hours summary
• Time cards
• Equipment requisitions
• Materials purchase orders
DESIGN
• Design specifications
• Materials list
OPERATIONS
• Task plans
• Employee/project time cards
• Task status
• Materials usage log
MANUFACTURING
CALIBRATIONS

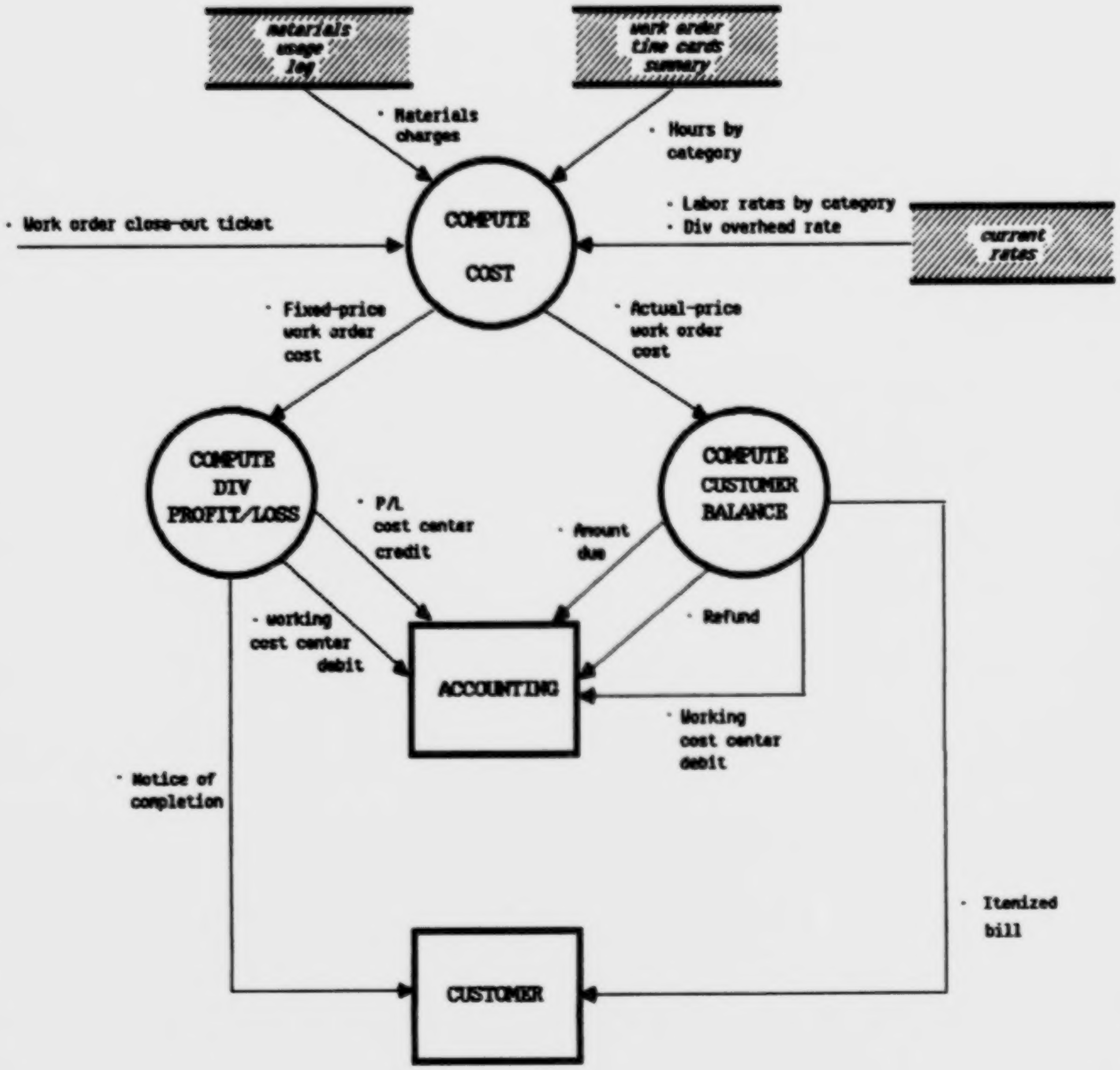
NOTES

OPERATIONS is responsible for coordinating the efforts of MANUFACTURING and CALIBRATIONS, scheduling tasks, ordering materials and equipment, reporting material and labor spent on each project.

INSTRUMENT FABRICATION DIVISION

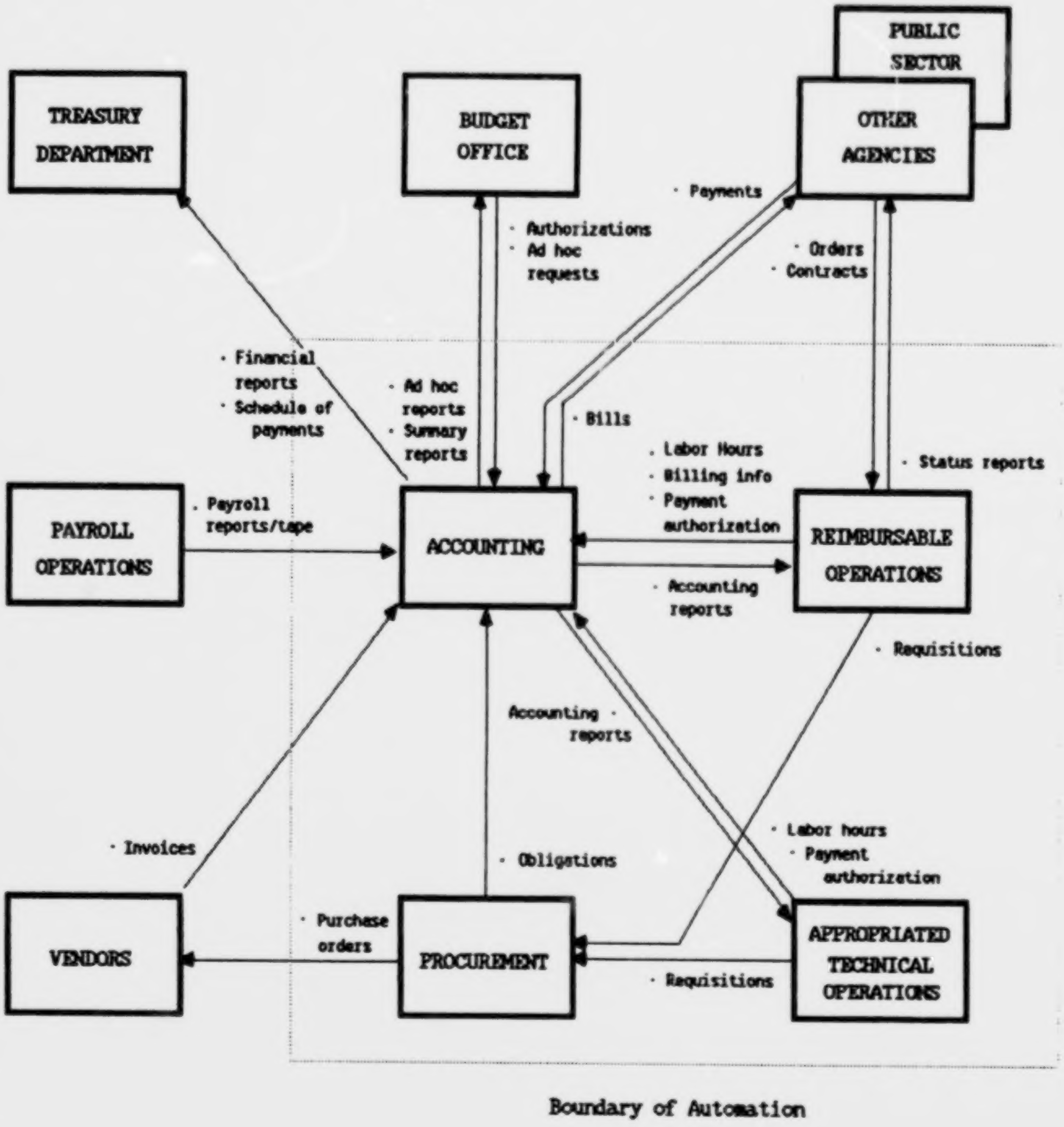
Local Information-flow Model

Function : Close Out Work Order


materials usage log
work order time cards summary
· Materials charges
· Hours by category
· Work order close-out ticket
COMPUTE COST
· Labor rates by category
· Div overhead rate
current rates
· Fixed-price work order cost
· Actual-price work order cost
COMPUTE DIV PROFIT/LOSS
COMPUTE CUSTOMER BALANCE
· P/L cost center credit
· Amount due
· working cost center debit
· Refund
ACCOUNTING
· Working cost center debit
· Notice of completion
· Itemized bill
CUSTOMER

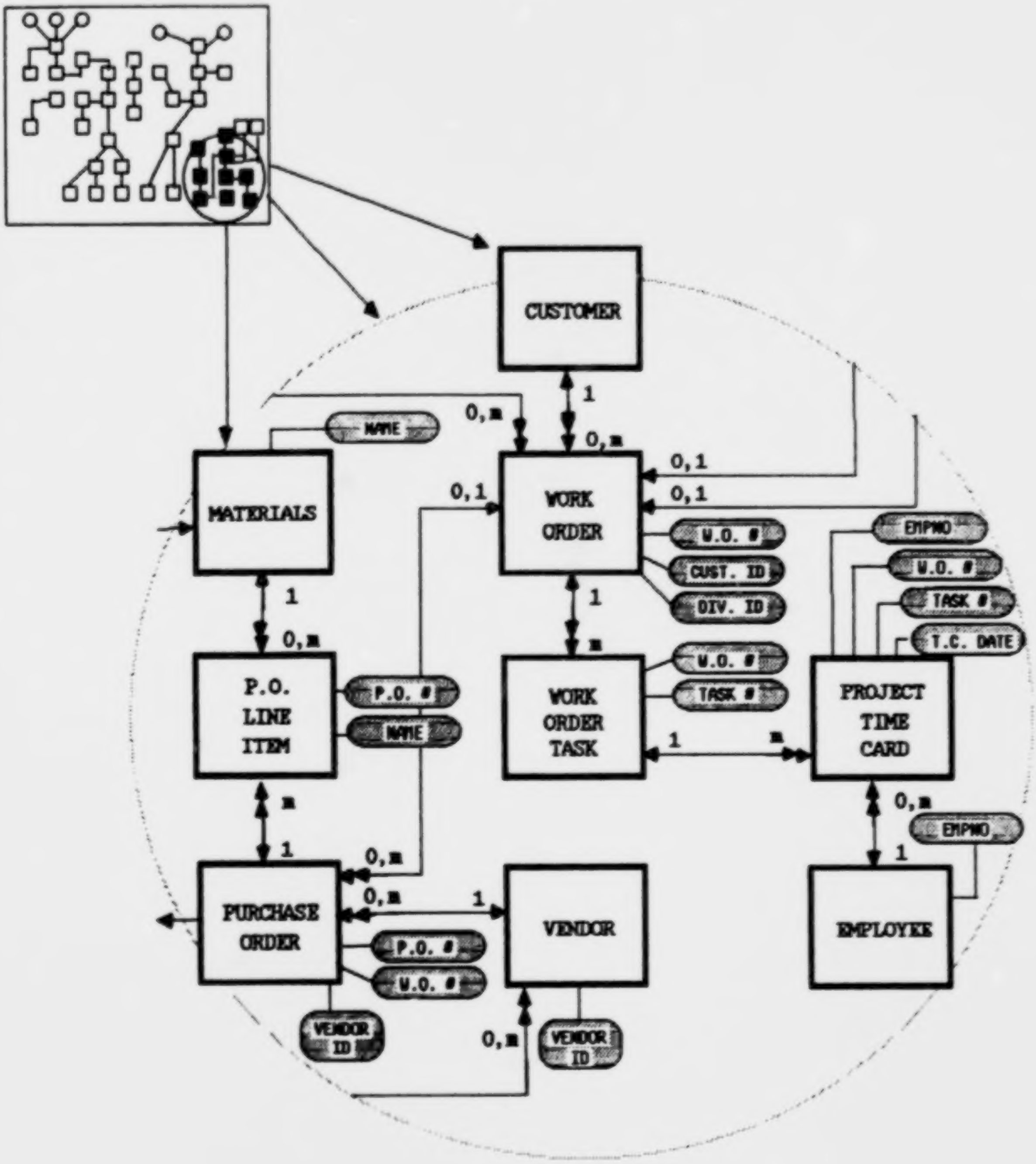
AGENCY FINANCIAL MANAGEMENT SYSTEM

Global Information-flow Model


PUBLIC SECTOR
TREASURY DEPARTMENT
BUDGET OFFICE
OTHER AGENCIES
· Payments
· Authorizations
· Ad hoc requests
· Orders
· Contracts
· Financial reports
· Schedule of payments
· Ad hoc reports
· Summary reports
· Bills
. Labor Hours
· Billing info
· Payment authorization
· Status reports
PAYROLL OPERATIONS
. Payroll reports/tape
ACCOUNTING
REIMBURSABLE OPERATIONS
· Accounting reports
· Requisitions
Accounting · reports
· Labor hours
· Payment authorization
· Invoices
· Obligations
VENDORS
· Purchase orders
PROCUREMENT
· Requisitions
APPROPRIATED TECHNICAL OPERATIONS
Boundary of Automation

AGENCY FINANCIAL MANAGEMENT SYSTEM

ENTITY-RELATIONSHIP DIAGRAM OF CONCEPTUAL SCHEMA

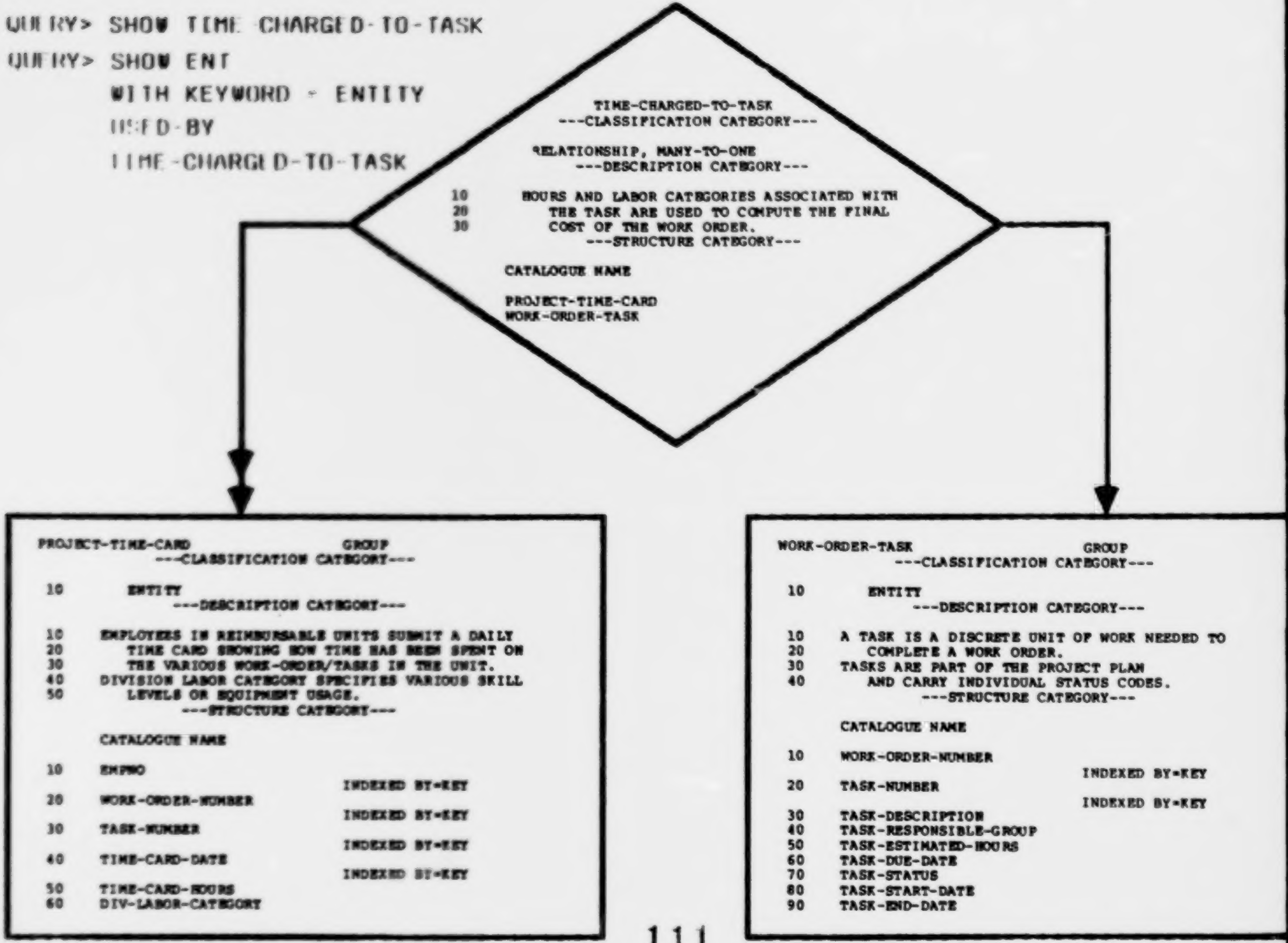

CUSTOMER
1
0,m
0,m
0,1
0,1
NAME
MATERIALS
0,1
WORK ORDER
W.O. #
CUST. ID
DIV. ID
1
0,m
1
m
EMPNO
W.O. #
TASK #
T.C. DATE
P.O. LINE ITEM
P.O. #
NAME
W.O. #
TASK #
WORK ORDER TASK
1
m
PROJECT TIME CARD
m
1
0,m
0,m
EMPNO
1
0,m
1
PURCHASE ORDER
VENDOR
EMPLOYEE
P.O. #
W.O. #
VENDOR ID
0,m
VENDOR ID

NOTES: Non-key attributes are not shown.
Data dictionary reports list all attributes.

Data Dictionary Display

Relationship : Time Charged To Task

```
QUERY> SHOW TIME-CHARGED-TO-TASK
QUERY> SHOW ENT
       WITH KEYWORD = ENTITY
       USED-BY
       TIME-CHARGED-TO-TASK
```



```
                TIME-CHARGED-TO-TASK
           ---CLASSIFICATION CATEGORY---

      RELATIONSHIP, MANY-TO-ONE
              ---DESCRIPTION CATEGORY---

10      HOURS AND LABOR CATEGORIES ASSOCIATED WITH
20         THE TASK ARE USED TO COMPUTE THE FINAL
30         COST OF THE WORK ORDER.
                ---STRUCTURE CATEGORY---

     CATALOGUE NAME

     PROJECT-TIME-CARD
     WORK-ORDER-TASK
```

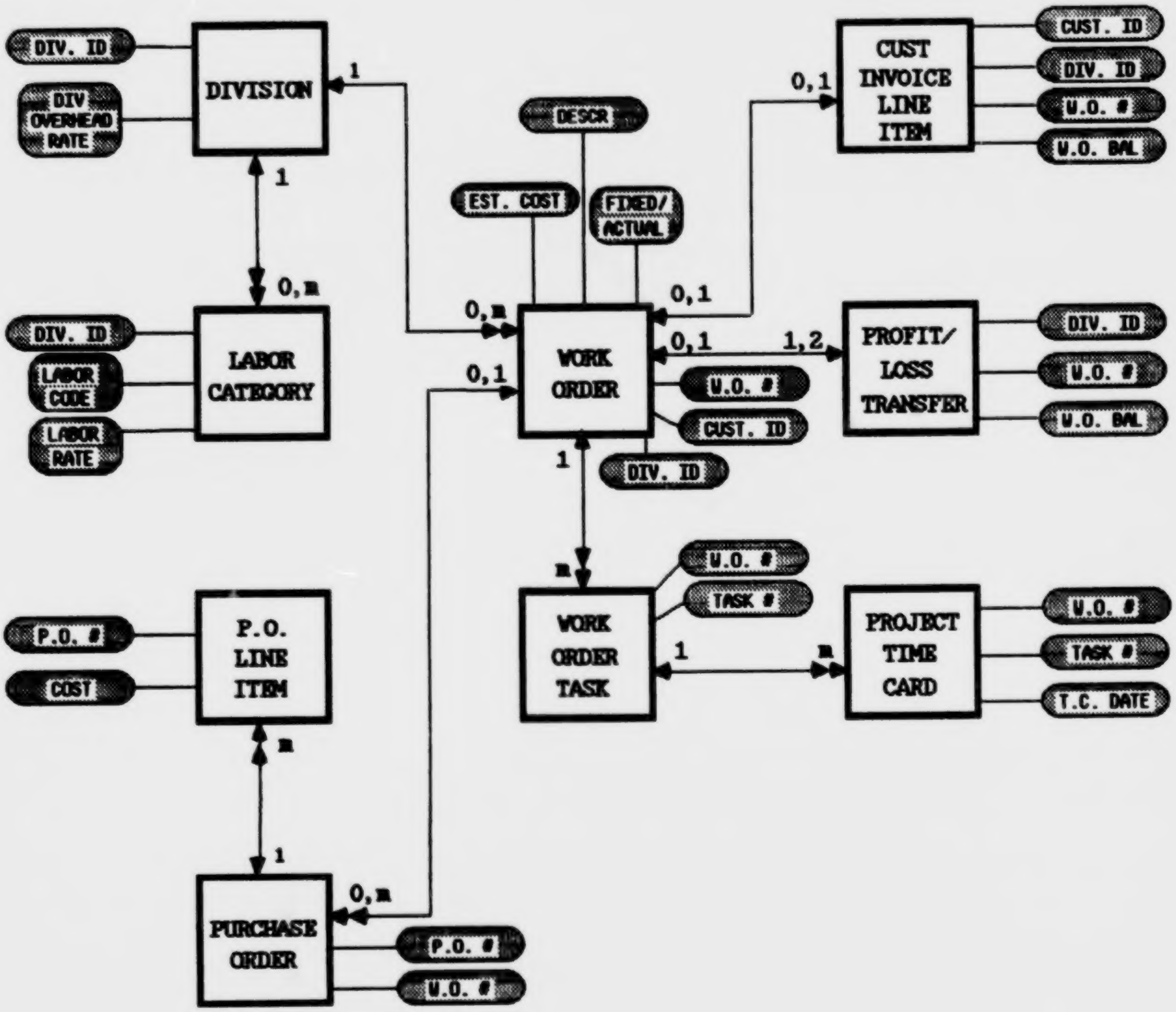
```
PROJECT-TIME-CARD                    GROUP
             ---CLASSIFICATION CATEGORY---

 10        ENTITY
               ---DESCRIPTION CATEGORY---

 10     EMPLOYEES IN REIMBURSABLE UNITS SUBMIT A DAILY
 20        TIME CARD SHOWING HOW TIME HAS BEEN SPENT ON
 30        THE VARIOUS WORK-ORDER/TASKS IN THE UNIT.
 40     DIVISION LABOR CATEGORY SPECIFIES VARIOUS SKILL
 50        LEVELS OR EQUIPMENT USAGE.
                ---STRUCTURE CATEGORY---

        CATALOGUE NAME

 10     EMPNO
                                     INDEXED BY-KEY
 20     WORK-ORDER-NUMBER
                                     INDEXED BY-KEY
 30     TASK-NUMBER
                                     INDEXED BY-KEY
 40     TIME-CARD-DATE
                                     INDEXED BY-KEY
 50     TIME-CARD-HOURS
 60     DIV-LABOR-CATEGORY
```

```
WORK-ORDER-TASK                         GROUP
             ---CLASSIFICATION CATEGORY---

 10        ENTITY
               ---DESCRIPTION CATEGORY---

 10     A TASK IS A DISCRETE UNIT OF WORK NEEDED TO
 20        COMPLETE A WORK ORDER.
 30     TASKS ARE PART OF THE PROJECT PLAN
 40        AND CARRY INDIVIDUAL STATUS CODES.
                ---STRUCTURE CATEGORY---

        CATALOGUE NAME

 10     WORK-ORDER-NUMBER
                                     INDEXED BY-KEY
 20     TASK-NUMBER
                                     INDEXED BY-KEY
 30     TASK-DESCRIPTION
 40     TASK-RESPONSIBLE-GROUP
 50     TASK-ESTIMATED-HOURS
 60     TASK-DUE-DATE
 70     TASK-STATUS
 80     TASK-START-DATE
 90     TASK-END-DATE
```

AGENCY FINANCIAL MANAGEMENT SYSTEM

EXTERNAL SCHEMA

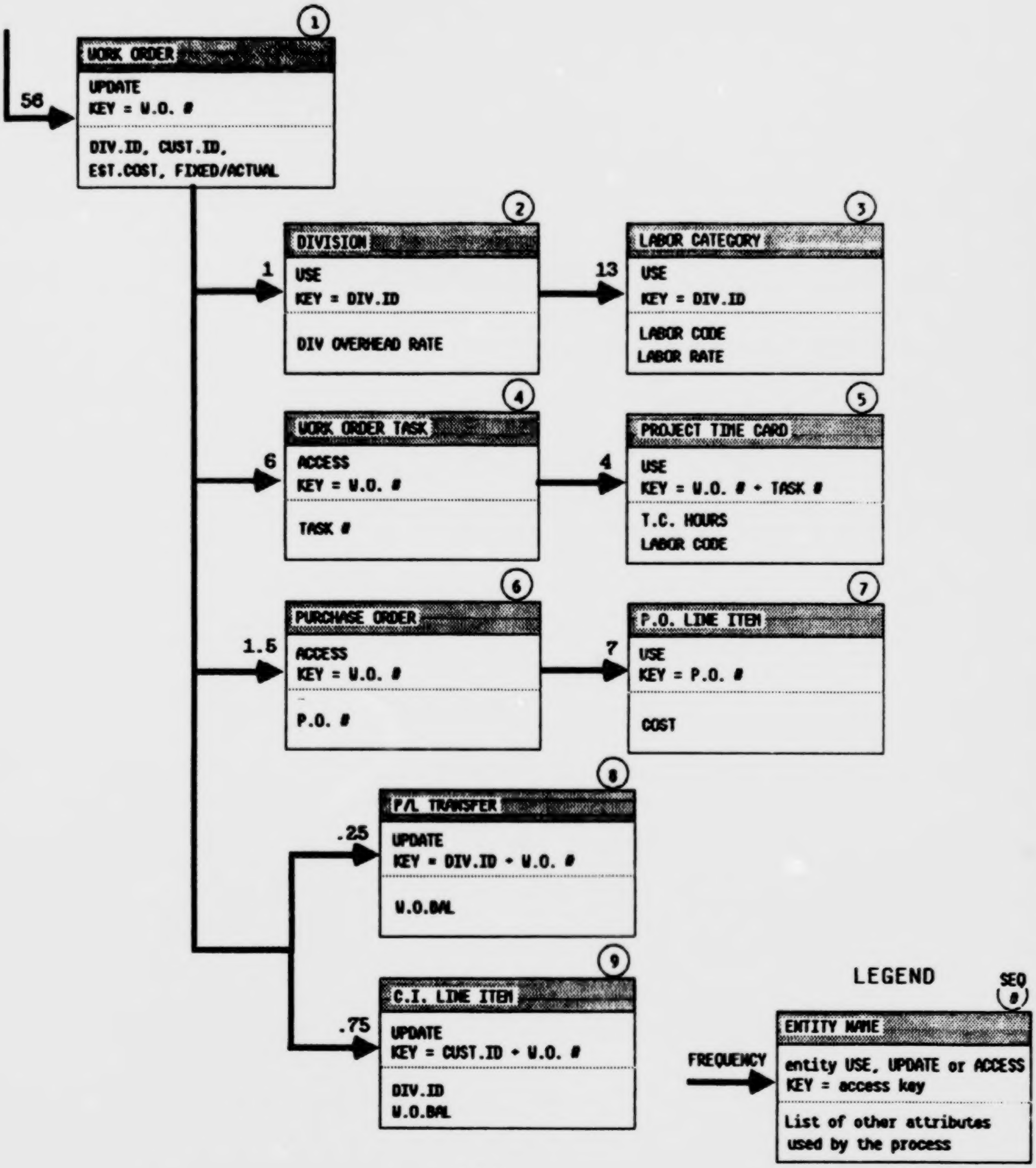
Function : Close Out Work Order


DIV. ID
DIV OVERHEAD RATE
DIVISION
1
1
0,m
DIV. ID
LABOR CODE
LABOR RATE
LABOR CATEGORY
P.O. #
COST
P.O. LINE ITEM
m
1
PURCHASE ORDER
0,m
P.O. #
W.O. #
DESCR
EST. COST
FIXED/ ACTUAL
0,m
0,1
WORK ORDER
W.O. #
CUST. ID
DIV. ID
1
m
W.O. #
TASK #
WORK ORDER TASK
1
m
PROJECT TIME CARD
W.O. #
TASK #
T.C. DATE
0,1
0,1
CUST INVOICE LINE ITEM
CUST. ID
DIV. ID
W.O. #
W.O. BAL
0,1
1,2
PROFIT/ LOSS TRANSFER
DIV. ID
W.O. #
W.O. BAL

NOTE : Entities, relationships and attributes not used by this function are not shown. Complete details are available from the data dictionary.

EXTERNAL SCHEMA "OVERLAY" WORKLOAD FOR FUNCTION

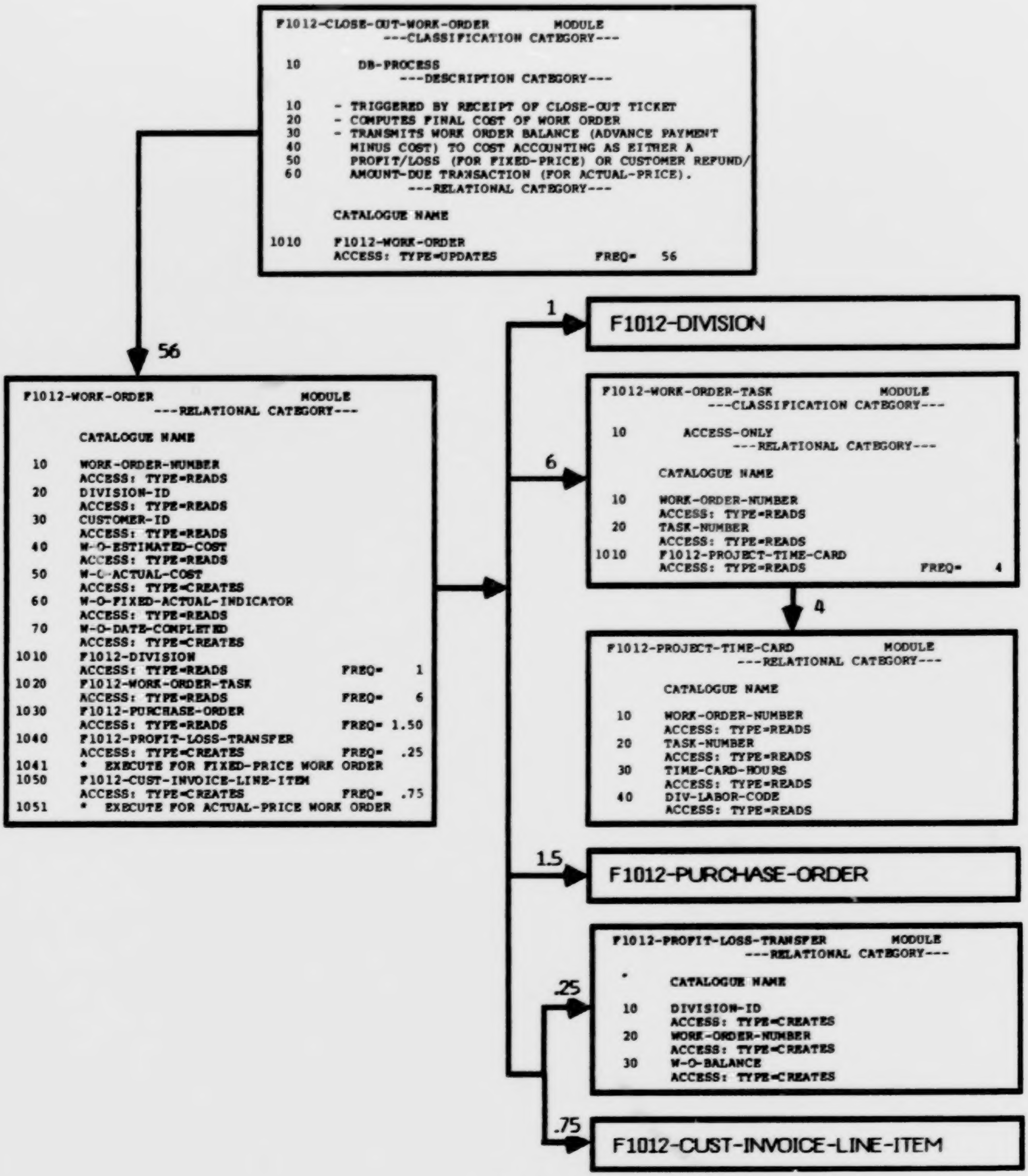
"Close Out Work Order"
Biweekly Statistics for All Reimbursable Divisions



①
WORK ORDER
UPDATE
KEY = W.O. #
DIV.ID, CUST.ID,
EST.COST, FIXED/ACTUAL
56
②
DIVISION
USE
KEY = DIV.ID
DIV OVERHEAD RATE
1
③
LABOR CATEGORY
USE
KEY = DIV.ID
LABOR CODE
LABOR RATE
13
④
WORK ORDER TASK
ACCESS
KEY = W.O. #
TASK #
6
⑤
PROJECT TIME CARD
USE
KEY = W.O. # + TASK #
T.C. HOURS
LABOR CODE
4
⑥
PURCHASE ORDER
ACCESS
KEY = W.O. #
P.O. #
1.5
⑦
P.O. LINE ITEM
USE
KEY = P.O. #
COST
7
⑧
P/L TRANSFER
UPDATE
KEY = DIV.ID + W.O. #
W.O.BAL
.25
⑨
C.I. LINE ITEM
UPDATE
KEY = CUST.ID + W.O. #
DIV.ID
W.O.BAL
.75
LEGEND
SEQ
#
ENTITY NAME
entity USE, UPDATE or ACCESS
KEY = access key
List of other attributes
used by the process
FREQUENCY

DATA DICTIONARY DISPLAY

WORKLOAD FOR FUNCTION



```
F1012-CLOSE-OUT-WORK-ORDER         MODULE
           ---CLASSIFICATION CATEGORY---

 10         DB-PROCESS
              ---DESCRIPTION CATEGORY---

 10      - TRIGGERED BY RECEIPT OF CLOSE-OUT TICKET
 20      - COMPUTES FINAL COST OF WORK ORDER
 30      - TRANSMITS WORK ORDER BALANCE (ADVANCE PAYMENT
 40        MINUS COST) TO COST ACCOUNTING AS EITHER A
 50        PROFIT/LOSS (FOR FIXED-PRICE) OR CUSTOMER REFUND/
 60        AMOUNT-DUE TRANSACTION (FOR ACTUAL-PRICE).
               ---RELATIONAL CATEGORY---

        CATALOGUE NAME

1010    F1012-WORK-ORDER
        ACCESS: TYPE=UPDATES              FREQ=   56
```

```
F1012-WORK-ORDER                   MODULE
             ---RELATIONAL CATEGORY---

        CATALOGUE NAME

  10    WORK-ORDER-NUMBER
        ACCESS: TYPE=READS
  20    DIVISION-ID
        ACCESS: TYPE=READS
  30    CUSTOMER-ID
        ACCESS: TYPE=READS
  40    W-O-ESTIMATED-COST
        ACCESS: TYPE=READS
  50    W-O-ACTUAL-COST
        ACCESS: TYPE=CREATES
  60    W-O-FIXED-ACTUAL-INDICATOR
        ACCESS: TYPE=READS
  70    W-O-DATE-COMPLETED
        ACCESS: TYPE=CREATES
1010    F1012-DIVISION
        ACCESS: TYPE=READS            FREQ=    1
1020    F1012-WORK-ORDER-TASK
        ACCESS: TYPE=READS            FREQ=    6
1030    F1012-PURCHASE-ORDER
        ACCESS: TYPE=READS            FREQ= 1.50
1040    F1012-PROFIT-LOSS-TRANSFER
        ACCESS: TYPE=CREATES          FREQ=  .25
1041    *  EXECUTE FOR FIXED-PRICE WORK ORDER
1050    F1012-CUST-INVOICE-LINE-ITEM
        ACCESS: TYPE=CREATES          FREQ=  .75
1051    *  EXECUTE FOR ACTUAL-PRICE WORK ORDER
```

```
F1012-WORK-ORDER-TASK              MODULE
          ---CLASSIFICATION CATEGORY---

 10         ACCESS-ONLY
              ---RELATIONAL CATEGORY---

        CATALOGUE NAME

 10     WORK-ORDER-NUMBER
        ACCESS: TYPE=READS
 20     TASK-NUMBER
        ACCESS: TYPE=READS
1010    F1012-PROJECT-TIME-CARD
        ACCESS: TYPE=READS                FREQ=    4
```

```
F1012-PROJECT-TIME-CARD            MODULE
             ---RELATIONAL CATEGORY---

        CATALOGUE NAME

 10     WORK-ORDER-NUMBER
        ACCESS: TYPE=READS
 20     TASK-NUMBER
        ACCESS: TYPE=READS
 30     TIME-CARD-HOURS
        ACCESS: TYPE=READS
 40     DIV-LABOR-CODE
        ACCESS: TYPE=READS
```

```
F1012-PROFIT-LOSS-TRANSFER         MODULE
             ---RELATIONAL CATEGORY---

 *      CATALOGUE NAME

 10     DIVISION-ID
        ACCESS: TYPE=CREATES
 20     WORK-ORDER-NUMBER
        ACCESS: TYPE=CREATES
 30     W-O-BALANCE
        ACCESS: TYPE=CREATES
```

INDENTED INDEX
EXTERNAL SCHEMA FOR FUNCTION

F1012-CLOSE-OUT-WORK-ORDER

```
RELATIVE LEVEL/DATA CATALOGUE NAME                 ENTRY TYPE   PAGE

. F1012-CLOSE-OUT-WORK-ORDER                       MODULE          2
. . F1012-WORK-ORDER                               MODULE          3
. . . WORK-ORDER-NUMBER                              ELEMENT       4
. . . DIVISION-ID                                    ELEMENT       5
. . . CUSTOMER-ID                                    ELEMENT       6
. . . W-O-ESTIMATED-COST                             ELEMENT       7
. . . W-O-ACTUAL-COST                                ELEMENT       8
. . . W-O-FIXED-ACTUAL-INDICATOR                     ELEMENT       9
. . . W-O-DATE-COMPLETED                             ELEMENT      10
. . . F1012-DIVISION                               MODULE         11
. . . . DIVISION-ID                                  ELEMENT      12
. . . . DIV-OVERHEAD-RATE                            ELEMENT      13
. . . . F1012-DIV-LABOR-CATEGORY                   MODULE         14
. . . . . DIVISION-ID                                ELEMENT      15
. . . . . DIV-LABOR-CODE                             ELEMENT      16
. . . . . DIV-LABOR-RATE                             ELEMENT      17
. . . F1012-WORK-ORDER-TASK                        MODULE         18
. . . . WORK-ORDER-NUMBER                            ELEMENT      19
. . . . TASK-NUMBER                                  ELEMENT      20
. . . . F1012-PROJECT-TIME-CARD                    MODULE         21
. . . . . WORK-ORDER-NUMBER                          ELEMENT      22
. . . . . TASK-NUMBER                                ELEMENT      23
. . . . . TIME-CARD-HOURS                            ELEMENT      24
. . . . . DIV-LABOR-CODE                             ELEMENT      25
. . . F1012-PURCHASE-ORDER                         MODULE         26
. . . . WORK-ORDER-NUMBER                            ELEMENT      27
. . . . PURCHASE-ORDER-NUMBER                        ELEMENT      28
. . . . F1012-PURCHASE-ORDER-LINE-ITEM             MODULE         29
. . . . . PURCHASE-ORDER-NUMBER                      ELEMENT      30
. . . . . P-O-LINE-ITEM-COST                         ELEMENT      31
. . . F1012-PROFIT-LOSS-TRANSFER                   MODULE         32
. . . . DIVISION-ID                                  ELEMENT      33
. . . . WORK-ORDER-NUMBER                            ELEMENT      34
. . . . W-O-BALANCE                                  ELEMENT      35
. . . F1012-CUST-INVOICE-LINE-ITEM                 MODULE         36
. . . . CUSTOMER-ID                                  ELEMENT      37
. . . . WORK-ORDER-NUMBER                            ELEMENT      38
. . . . DIVISION-ID                                  ELEMENT      39
. . . . W-O-BALANCE                                  ELEMENT      40

*** E N D   O F   I N D E X   ***
```

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NBS/SP-500/122	2. Performing Organ. Report No.	3. Publication Date February 1985

4. TITLE AND SUBTITLE

Computer Science and Technology:
Guide on Logical Database Design

5. AUTHOR(S)

Elizabeth N. Fong, Margaret W. Henderson, David K. Jefferson, Joan M. Sullivan

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Same as in item 6 above.

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 85-600500

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This report discusses an iterative methodology for Logical Database Design. The methodology includes four phases: Local Information-flow Modeling, Global Information-flow Modeling, Conceptual Schema Design, and External Schema Modeling. These phases are intended to make maximum use of available information and user expertise, including the use of a previous Needs Analysis, and to prepare a firm foundation for physical database design and system implementation. The methodology recommends analysis from different points of view--organization, function, and event--in order to ensure that the logical database design accurately reflects the requirements of the entire population of future users. The methodology also recommends computer support from a data dictionary system, in order to conveniently and accurately handle the volume and complexity of design documentation and analysis. The report places the methodology in the context of the complete system life cycle. An appendix of illustrations shows examples of how the four phases of the methodology can be implemented.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

data dictionary system; data dictionary system standard; data management; data model; database design; database management system, DBMS; Entity-Relationship-Attribute Model; Information Resource Dictionary System, IRDS; logical database design.

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

115

15. Price

USCOMM-DC 6043-P80

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ______________________________

Company ______________________________

Address ______________________________

City ______________ State ____________ Zip Code ________

(Notification key N-503)

*U.S. GOVERNMENT PRINTING OFFICE : 1985 O-461-105/10198

NBS Technical Publications

Periodicals

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.